Seconde ére : le logiciel & les éditeurs

Lors de l'ére du matériel détaillée dans la section précédente, les acteurs dominants étaient les constructeurs capables d'assembler ces machines compliquées, exigeantes et horriblement coûteuses. Le logiciel n'est alors perçu que comme une contrainte nécessaire, pas comme la base de la valeur ajoutée proposée aux clients !
D'ailleurs, le logiciel n'était même pas vendu en tant que tel (il est compris dans le prix de l'ordinateur) et la notion d'éditeurs de logiciels mettra longtemps à être inventée (par des pionniers comme Computers Associates).

Le logiciel prend la place centrale
L'ère du logiciel et des éditeurs commence vraiment grâce et avec le PC.
Avec le PC, tout change, c'est le logiciel qui devient la pièce centrale, la raison qui fait acheter des PC par palettes pour en équiper TOUS les employés d'une entreprise. Lotus 123 (un tableur performant) est l'élément qui fit basculer son époque (la poussée finale d'un tournant entamé depuis l'Apple II et Visicalc, le premier PC et le premier tableur).
L'industrie de la grande informatique, dans son ensemble, n'était pas consciente de l'importance de la micro informatique et des bouleversements que l'introduction du PC allait amener. À part quelques avant-gardistes comme Bruno Lussato, la plupart des observateurs n'imaginaient pas une seconde que les micro-ordinateurs reliés en réseaux pourraient représenter une architecture alternative aux grands mainframes.
Donc, la plupart des grands constructeurs vont mollement suivre la tendance et proposer eux aussi leur PC (le plus à la traîne sera DEC handicapé par son président fondateur, Ken Olsen, qui avait déclaré « le PC est un jouet »…), mais sans trop d'efforts et sans vraiment y croire.
La principale explication à cette inertie réside dans la structure de ces organisations qui reposaient toutes sur le principe de la vente directe de produits très haut de gamme avec une armée de commerciaux payés royalement qui avaient pour mission de placer des machines très coûteuses et à forte marge. Dans ce schéma, il n'y avait rien à gagner à vendre en direct des PC abordables et à faible marge, seule la vente indirecte permettait de s'y retrouver.

Le passage à l'ère du logiciel a eu d'autres conséquences, pas toujours vécues consciemment par ses promoteurs. Ainsi, le PC a considérablement élargi le cercle des gens concernés par l'informatique : des utilisateurs évidemment mais aussi les professionnels qui sont devenus bien plus nombreux.
Auparavant confiné aux applications d'entreprises, l'ordinateur s'ouvre alors aux besoins personnels et adresse vraiment la cible individuelle.
C'est ici une des clés pour comprendre pourquoi le succès du PC a été aussi fort et pourquoi l'engouement a été général : la micro a permis l'élargissement des usages de l'informatique. L'informatique sur mainframe ne s'intéressait et ne permettait que les traitements de masse, l'utilisateur individuel était quasiment nié, les applications ne s'adressaient qu'à l'organisation dans son ensemble et les besoins particuliers (telle la production de documents comme le courrier) étaient quasiment négligés (il y avait bien quelques systèmes de time-sharing qui servaient des utilisateurs individuels et les minis servaient souvent des besoins locaux dans l'organisation).
Certains acteurs comme Wang ont su profiter de cette lacune et ont proposé des solutions adressant correctement tel ou tel domaine d'application (le traitement de texte en particulier dans le cas de Wang) mais sur des systèmes propriétaires et pour lesquels les développements tiers n'étaient pas encouragés (toujours cette logique de l'intégration verticale).
Le PC a submergé des acteurs spécialisés comme Wang grâce à son côté versatile. Sur le plan du traitement de texte, les programmes sur PC (WordStar, Wordperfect, MS Word) n'étaient pas toujours du même niveau que la solution de Wang mais, à la différence de ce dernier, le PC n'était pas limité à cette seule application…
Un marché de professionnels qui devient un marché de masse
C'est la richesse de l'offre logicielle qui a déclenché l'emballement. L'invention du tableur et sa large disponibilité à travers le PC ont provoqué un ralliement en masse de toute

une population qui était jusque là restée à l'écart de l'informatique traditionnelle. Pagemaker (et l'imprimante laser abordable) va faire de même dans le monde de l'édition. C'est une constante de la mise à disposition de nouveaux usages, elle provoque l'apparition et le ralliement de nouvelles franges d'utilisateurs. De loin en loin, les rangs grossissent toujours plus et le marché ainsi constitué devient vite le marché dominant, celui où il faut être présent, celui qui a la dynamique, celui qui rend les autres obsolètes !
Dans un premier temps, la micro-informatique n'a pas été relayée par les services informatiques internes des entreprises parce qu'elle n'offrait que peu de prises aux prestations habituellement assurées par ces services. D'ailleurs, l'irruption de la micro a même suscité la création de services spécialisés en marge de l'organisation de l'informatique traditionnelle au sein des entreprises. Et culturellement, la mise en place de minis puis de micros était même souvent une véritable révolte contre le service informatique central...
Ce n'est que progressivement que les deux (micro-informatique et informatique classique) se sont rejointes sur la nécessité de cohabiter afin que l'utilisateur puisse accéder aux applications du mainframe sans être placé dans la situation ubuesque d'avoir deux écrans sur son bureau… Cet élargissement sans précédent des usages et des populations concernées acheva de concrétiser le rôle de l'informatique dans le fonctionnement interne de l'entreprise.

Pour comprendre comment et pourquoi on à ce qui était tout juste une notion annexe de la "grande informatique" à des produits s'adressant au marché de masse vendus par des nouveaux acteurs (les éditeurs) qui ont renvoyé les contructeurs traditionnels dans l'ombre, il faut retracer l'évolution du logiciel et revenir sur le contexte informatique de la fin des années 60...

Du logiciel aux éditeurs, une lente évolution...

Le point de départ : l'unbundling
Dans cette évolution, la décision d'IBM connue sous le nom de "unbundling" (dégroupage) est souvent considérée comme le point de départ du développement de l'industrie du logiciel. En janvier 1969, le département de la justice a engagé des poursuites contre IBM en application de la loi antitrust. Pour faire face à cette poursuite, IBM pris les devants et l'unbundling a été annoncé par IBM en juin 1969 : moyennant une réduction de 3 % du prix du matériel, les logiciels -ainsi que la majeure partie de la formation et de l'assistance technique- seraient vendus désormais à part.
Il est évident que cette décision a représenté un facteur d'influence important mais les progiciels existaient et étaient commercialisés avant ce tournant. De plus, l'application de la décision d'IBM pris quelques années, particulièrement en Europe où les filiales de big blue n'ont vraiment commencé à séparer la facturation du matériel et du logiciel qu'à partir de 1972.
A partir de cette séparation, le logiciel est effectivement devenu un secteur commercial et a pu prendre son envol. Cependant, les produits logiciels indépendants (des contructeurs) existaient avant l'unbundling.
Autoflow (mis sur le marché par Applied Data Research en 1965) et Mark IV (lancé en 1967 par Informatics) sont considérés comme les deux premiers progiciels de l'histoire. Si ce n'est sans doute pas vraiment les tous premiers, ils sont au moins considérés comme les plus influents sur ce marché naissant.
C'est le dégroupage qui donna le signal de la croissance pour les progiciels. Avant cela, ADR n'avait vendu que 300 exemplaires d'Autoflow, après cela, tout change : ADR pu vendre 100 exemplaires de son logiciel dès le premier trimestre de 1969 et la société connu une croissance de 20% dans les années qui suivirent…
Même phénomène chez Informatics avec Mark IV : les prévisions de ventes étaient timides avant l'unbundling puisque la société espérait $1,7 millions pour l'exercice 71 alors qu'elle fit $2,8 millions dès 1969...

L’édition de progiciels n’a pas été traitée avec l’enthousiasme qu’on pourrait imaginer par les grandes SSII de l’époque pour plusieurs raisons : tout d’abord, dans le domaine du logiciel, les aptitudes ou compétences particulières qui faisaient qu'une société réussissait sur un segment de marché devenaient des faiblesses ou même des handicaps dans un autre. C'est la principale raison qui fait que très peu de SSII sont parvenues à déborder de leur marché initial (les services) et que les éditeurs sont vraiment une catégorie à part.

Un exemple avec une SSII très connues dans les années 60 : ADP. ADP (Automatic Data Processing) était la société la plus connue aux USA dans le traitement des salaires de ces clients. En 1970, elle traitait les payes de 7000 sociétés clientes, totalisant 5 milliards de dollars de salaires. Le traitement à façon, c'était cela. L'infogérance pouvait être considérée comme l'inverse du traitement à façon : une société d'infogérance ne fournissait pas un traitement informatique dans ses propres locaux (et sur ces propres ordinateurs), elle gérait l'informatique en fait les systèmes informatiques du client chez lui.
L'infogèrance s'était déjà pratiquée longtemps avant l'existence des ordinateurs. Dans le cadre de la défense américaine, le prestataire d'infogérance fournissait du personnel et des compétences pour exploiter une installation technique au sein de l'organisation du client qui en restait normalement propriétaire et responsable.
Cependant, c'est Ross Perot qui a introduit cette notion d'infogérance dans l'informatique avec sa société EDS fondée en 1962. La stratégie de Perot consistait à repérer des grandes installations informatiques mal exploitées et il proposait alors qu'EDS s'en occupe pour un coût fixe, au forfait.
La seconde raison est que les coûts de commercialisation des progiciels se sont révélés être plus importants que prévu...

La crise du logiciel : des systèmes de plus en plus gros
Avec le lancement de chaque nouveau système au début des années 60, le logiciel système fournit par IBM pour accompagner ses ordinateurs augmentait -en taille- d'un facteur 10 tous les cinq ans !
On a estimé que la part du coût de développement des logiciels de base est passé de 10% en 1960 à 40% en 1965 sur le total du coût de mise en oeuvre d'un nouveau système...

Le concept de logiciel prêt à l'emploi fut d'abord une réponse technique à la "crise du logiciel" de la part des constructeurs : le nombre d'installations de systèmes croissait bien plus vite que le nombre de programmeurs formés, les progiciels étaient un moyen évident de réduire ce goulot d'étranglement (on passe donc du logiciel "sur-mesure" pour un seul client à un logiciel "prêt à l'emploi" pour le plus grand nombre d'abord pour faire face à la demande). Les SSII présentes sur le secteur informatique des années 60 se rendirent vite compte que les progiciels "gratuits" des constructeurs ne suffiraient pas à satisfaire la demande des clients qui croissait rapidement. Cela représentait une opportunité de croissance tout à fait intéressante mais, en se lançant sur ce nouveau marché, les SSII se rendirent compte que les coûts de commercialisation de ces progiciels étaient encore bien plus élevés que les coûts de fabrication (conception, programmation, test et duplication) qui étaient déjà élevés. Ce business model (celui des progiciels) était radicalement différent de celui du service auquel les SSII étaient évidemment habituées et elles s’étaient déjà structurées autour.

Durant les années 70, le développement d'un système de base de données (une des premières cibles de cette époque où tout manquait encore) demandait entre $500 000 et $1 million et qu'il fallait jusqu'à cinq ans pour en vendre 500 exemplaires (avec un prix unitaire évoluant entre $25 000 et $50 000). En conséquence, les premiers progiciels étaient vendus comme les systèmes qui les faisaient tourner : avec des équipes de commerciaux prospectant sur le terrain avec pratiquement aucun effet d'échelle. Les coûts de commercialisation étaient donc bien plus importants qu'escomptés au départ et

cela eu pour effet de refroidir (voire de doucher complétement) l'envie des SSII de se lancer dans ce secteur...

De plus, les logiciels pour mainframes (et en particulier pour mainframes IBM) ont bénéficié d'un élément inattendu : la longévité de ces derniers. IBM avait prévu de remplacer ses systèmes 360/370 par une gamme complètement nouvelle (et incompatible avec les précédentes) : Future System. Mais cet objectif ambitieux à fait long feu : le projet commencé en septembre 1971 a été abandonné en 1975 car il présentait des performances décevantes et que les études montraient que sans une voie de migration facile à suivre, les clients n'étaient pas prêts à sauter le pas vers des mainframes qui imposaient de tout redévelopper…
Les conséquences de ce non-événement ont été considérables : le marché des mainframes s'est figé durablement, les progrès se sont ralentis et les logiciels dédiés à ces machines ont eu un cycle de vie bien plus long que quiconque pouvait l'imaginer lorsqu'ils ont été conçus (ce qui a naturellement conduit à la déchirante révision connue sous le nom de "bug de l'an 2000" quelques décennies plus tard…).

=========
Encadré sur le projet "Future Systems" d'IBM (extraits issus de http://fr.wikipedia.org/wiki/Projet_FS)

Le projet Future Systems (FS) est un projet de recherche et développement entrepris par IBM au début des années 1970. Il visait à mettre sur le marché une ligne complète de produits informatiques nouveaux exploitant les techniques récentes pour simplifier fortement le développement des applications informatiques - afin essentiellement d'en accélérer la croissance. Ce projet a été abandonné en 1975.

Au début des années 70, IBM était face à un problème stratégique : si les coûts des matériels informatiques (y compris ceux produits par la compagnie) étaient en diminution constante, les coûts de programmation et d'exploitation, allaient eux, en augmentation. En conséquence, la seule amélioration du rapport performance/coût des machines ne répondait pas directement aux besoins futurs des clients. Avec une nouvelle architecture réduisant les coûts de développement et d'exploitation dans ses produits futurs, IBM améliorerait sa propre compétitivité tout en présentant des produits plus attractifs et donc plus aisément vendables.

En mai-juin 1971, une "task force" internationale fut réunie à Armonk par John Opel, alors vice-président d'IBM. Sa tache était d'explorer la faisabilité d'une nouvelle ligne d'ordinateurs. Ceux-ci exploiteraient le savoir-faire d'IBM pour rendre obsolète la génération précédente - les compatibles, mais pour le coup aussi ceux d'IBM. Cette "task force" conclut que le projet pouvait être accepté par le marché, mais - sans surprise - uniquement s'il permettait une réduction massive des coûts de développement, exploitation et maintenance des logiciels d'application.
Au vu de ces conclusions, la direction d'IBM décida d'entreprendre le projet "Future Systems" (FS), en lui assignant les objectifs suivants :
- rendre obsolètes tous les ordinateurs existants, y compris ceux d'IBM, en exploitant pleinement les techniques du moment [réf. nécessaire],
- offrir une réduction forte des coûts de développement et d'exploitation des applications,
- fournir une base techniquement solide au regroupement d'une part aussi élevée que possible des offres d'IBM (matériels, logiciels et services).

Lancement du projet

Le projet Future Systems (FS) fut lancé officiellement en Septembre 1971, à la suite des recommandations de la "task force" du deuxième trimestre 1971. Pendant toute sa durée, le projet FS se déroula dans des conditions de sécurité et de confidentialité très strictes bien qu'il mobilisa jusqu'à 2500 personnes. Le projet était divisé en nombreux sous-projets affectés à des équipes différentes. La documentation était découpée de la même façon en nombreux éléments, et l'accès à chaque document était soumis à vérification de la réalité du besoin réel par un organisme central du projet. Chaque document faisait l'objet d'un suivi et pouvait être rappelé à tout moment.
Une conséquence était que la plupart des personnes qui travaillaient sur le projet n'en avaient qu'une vue extrêmement fragmentaire, limitée à ce qu'ils devaient en savoir pour produire la contribution qu'on attendait d'eux. Certaines équipes travaillaient même pour le projet sans le savoir. Cela explique pourquoi, quand ils sont amenés à définir ce qu'était FS ou pourquoi le projet a été abandonné, la plupart des gens donnent une réponse très partielle, qui ne mentionne que la partie du projet FS qui entre dans leur domaine de compétence.

Trois implémentations de l'architecture FS étaient prévues : le modèle de haut de gamme était conçu à Poughkeepsie (état de New York), où étaient produits les ordinateurs les plus puissants d'IBM's ; le modèle de milieu de gamme était conçu à Endicott (état de New York), qui était responsable des ordinateurs de milieu de gamme ; le plus petit modèle était conçu à Rochester (Minnesota), qui avait la responsabilité des ordinateurs d'entrée de gamme.
Une gamme continue de performances pouvait être obtenue en faisant varier le nombre de processeurs par système à chacun des trois niveaux d'implémentation.

Un principe essentiel de FS était celui de la mémoire à un seul niveau ("single-level store") qui étendait l'idée de mémoire virtuelle à la totalité des données, temporaires ou persistantes, et rendait invisible au programmeur une quelconque différence entre accès à une base de données, à un fichier ou à un objet en mémoire.
Le réalisation de ce principe demandait que le mécanisme d'adressage, qui est au cœur de la machine, incorpore un système complet de gestion de la hiérarchie des mémoires et des parties importantes d'un système de gestion de bases de données, qui jusque là étaient réalisées sous forme de logiciels extérieurs à la machine proprement dite.
Un autre principe était l'utilisation d'instructions de haut niveau très complexes réalisées sous forme de microcode. Par exemple, certaines instructions étaient conçues pour supporter les structures de données et les opérations des langages de programmation de haut niveau tels que FORTRAN, COBOL et PL/I. En réalité, FS était conçu comme l'ordinateur à jeu d'instruction complexe (CISC) ultime.
Le plan d'ensemble prévoyait aussi un "contrôleur universel" pour traiter principalement les opérations d'entrée-sortie en dehors du processeur principal. Ce contrôleur universel devait avoir un jeu d'instructions très limité, réduit aux instructions nécessaires aux entrées-sorties. Il annonçait ainsi le concept d'ordinateur à jeu d'instructions réduit (RISC).
Sur cette idée, John Cocke, un des principaux concepteurs des premiers ordinateurs IBM, lança un projet de recherche destiné à concevoir le premier ordinateur RISC. Par la suite, l'architecture RISC, qui au sein d'IBM évolua vers l'architecture Power et PowerPC, devait se révéler beaucoup moins chère à réaliser et capable de cadences d'horloge bien supérieures.

Les raisons de l'abandon

Les raisons de l'abandon du projet en 1975 sont mal connues. A cause du cloisonnement mis en place pour préserver la sécurité, les raisons invoquées dépendent de l'interlocuteur, qui met en avant les difficultés rencontrées dans le domaine avec lequel il est familier.

Quelques raisons citées sont:
- les performances médiocres comparées à celles des systèmes 370 et compatibles.
- la complexité du jeu d'instructions, qui était jugé "incompréhensible" par de nombreux développeurs d'IBM.
- le coût pour les clients de la migration depuis leurs anciens systèmes. En effet, afin de laisser aux architectes le maximum de liberté pour définir un système vraiment révolutionnaire, la facilité de migration n'avait pas été incluse dans les objectifs initiaux, mais devait être traitée après coup par les aides logicielles.

Vers la fin du projet, il apparaissait que le coût de migration de la masse des investissements des clients dans des applications écrites en assembleur et en COBOL serait souvent supérieur au coût d'acquisition d'un nouveau système. De plus, la modification des habitudes s'avérait telle que la coexistence obligatoire pendant plusieurs années des deux systèmes serait une source de coûts énormes pour le constructeur comme pour ses clients.
En réalité, la réussite du projet nécessitait un grand nombre de percées dans tous les domaines, allant de la conception et de la fabrication des circuits jusqu'au marketing et à la maintenance, en passant par l'architecture et toutes les formes de logiciel. S'il était concevable que chacun des problèmes pris isolément soit résolu, la probabilité qu'ils puissent tous l'être dans les délais et de façon mutuellement compatible était pratiquement nulle, surtout compte-tenu des nombreux débats internes quant aux différentes voies de solution envisageables.
Les développements de FS furent plus onéreux que prévu. Des concepts qui nous sont aujourd'hui familiers n'existaient alors même pas, et les découvrir a impliqué beaucoup d'essais, de fausses pistes et de tâtonnements.

Les retombées du projet FS au sein d'IBM et au-delà
Pour les 370, les retombées les plus immédiates furent :
- une imprimante à laser nommée 3800 (les préfixes étaient en 38 parce qu'on supposait que la nouvelle série allait se nommer 380), qui avait des caractères programmables.
- un robot de stockage « à nid d'abeille » nommé 3850, qui était un système de stockage magnétique ne demandant strictement aucune présence humaine et pouvant fonctionner tout au long de l'année.
- des terminaux à caractères programmables qui furent d'abord le 3278 PS (monochrome), puis le 3279 PS (couleur) et pouvaient afficher des graphiques de gestion et autres.
- le logiciel GDDM (Graphical Data Display Manager);
- le migrateur automatique de fichiers HFS (Hierarchical File System), qui fut porté sous MVS.

Un système d'entrée de gamme inspiré du FS fut le Système 38, qui déconcerta un peu par ses concepts inhabituels, mais fut suivi d'un successeur nommé AS/400 qui connut un succès plus fort qu'attendu, et resta jusqu'à la fin des années 1990 un des fers de lance d'IBM. Seul problème : la gamme devait être étendue par le haut, les utilisateurs d'AS/400 n'étant pas enthousiastes du tout pour repasser au 370 et à ses successeurs.
=========

Ensuite, 3ème raison, les grandes SSII de l'époque avait une autre "grande idée" en vue qui contribuait à les distraire de l'édition de progiciels : le computer utility.

Le mirroir aux alouettes du "computer utility"
Le domaine dans lequel les SSII des années 60 se sont beaucoup investies était le "computer utility" ou "service bureau" qui était très à la mode à la fin des années 60 (nous avons déjà évoqué cet épisode dans "l'ère des constructeurs").
CSC, une des plus grosses SSII américaines, se lança dans ce secteur avec le projet "Infonet". Estimé à 50 millions de dollars en 1968, le budget d'Infonet avait enflé à 100 millions de dollars en 1970.
Infonet, une fois en exploitation, ne rapporta aucun bénéfice jusqu'en 1974 où ses ventes atteignirent 50 millions de dollars, loin des 300 millions annuels que CSC avait espéré en démarrant ce projet… Durant ce désastre, CSC survécut grâce à la production de logiciels sous contrat, son coeur de métier initial.
Cet échec cuisant (et CSC ne fut pas la seule SSII à se laisser prendre à ce "miroir aux alouettes" du computer utility) explique pourquoi la plupart des grandes SSII étaient ensuite réticentes à sortir de leur coeur de métier et à tenter l'aventure de l'édition de logiciels (en dehors des progiciels spécifiques que le hasard mettait entre leurs mains de temps en temps, suite à un projet mené avec un client qui débouchait ensuite sur une commercialisation).

Même une fois que le mirage du computer utility se soit dissipé, le secteur du progiciels pour entreprise restait encore relativement restreint et très morcellé. Entre 1977 et 1982, le nombre de sociétés américaines spécialisées dans ce domaine était tout de même passé de 600 à 1800. Dans le lot, il y avait quelques grandes entreprises mais même MSA, la principale société du secteur avec des ventes de $96 millions ne détenait que 2% du marché. Par comparaison à la même époque, IBM détenait plus de 50% du marché des mainframes où seuls une vingtaine de concurrrents se partageait ce domaine.
Le marché du progiciel n'était pas encore un marché de masse. En 1984, le best-seller des logiciels d'entreprises était Mark IV et il culminait à 3000 installations alors qu'à la même époque, Wordstar (un traitement de texte, le plus vendu des logiciels pour micro-ordinateur, secteur qui était alors en plein croissance) avait déjà été diffusé à 700 000 exemplaires. Mais l'installation de Mark IV coûtait $100 000 alors que la boite de Wordstar s'achetait à $495… Cette différence de prix explique une différence saillante entre les deux secteurs : le premier devait être vendu (et nécessitait même un effort commercial important) alors que le second était acheté par sa cible et devait juste être disponible dans les canaux de distributions qui alimentaient le marché.

Le marché du mainframe ne représentait pas assez de débouchés pour vraiment faire décoller la notion de progiciels mais, heureusement, la vague des mini-ordinateurs a été le relais que cette industrie embryonaire attendait...

Les intégrateurs
Avec la vague des minis est apparue aussi celle des intégrateurs. Les intégrateurs étaient principalement des éditeurs de logiciels qui pour faciliter le placement de leurs produits proposaient des systèmes complets "prêts à l'utilisation" avec le support et service après-vente nécessaires. L'exemple le plus célèbre est sans doute Computervision. Computervision commença par intégrer ses logiciels sur les minis de Data General puis utilisa des stations Sun (le contrat avec Computervision aida grandement Sun à décoller au début de son parcours…). L'autre intégrateur significatif était sûrement ASK avec ManMan.

ASK et le tout premier MRP
ASK computer systems, créé en 1972 par Sandra Kurtzig, était l'éditeur de ManMan, un progiciel de gestion de production (MRP pour Material Ressources Planning). L'histoire d'ASK et de ManMan est significative des méandres par lesquelles est passée l'évolution

commerciale des progiciels avant leur explosition et généralisation sur le marché, entrainant ainsi l'ère des éditeurs.
Tout d'abord, Sandra Kurtzig travaillait chez General Electric dans le département du time-sharing (c'était encore en plein dans le boum éphémère du "computer utility" que l'on a vu dans la section précédente). Elle eu l'occasion d'y développer un programme de MRP rudimentaire à la demande d'un client. A l'occasion de ce projet, elle pu se rendre compte du potentiel de ce type de logiciel et décida de se lancer en solo. Mais à cette époque, le coût de commercialisation des logiciels (avec des équipes commerciales nombreuses et des cycles de vente très longs) représentait une barrière d'entrée élevée, surtout quand on était seule et sans capital !
Pour contourner cette barrière, Sandra eu l'idée de distribuer le programme qu'elle venait de développer via le service Tymshare (encore un service de time-sharing qui survécu jusqu'en 1984 et son rachat par McDonnell Douglas pour être démantelé par la suite…). Passer par un service de Time-sharing était logique pour Sandra puisque c'est dans un de ces services (chez G&E) qu'elle avait commencé sa carrière. Cette idée fonctionna bien et Sandra pu développer son affaire. Bien au courant de ce qui se passait dans son domaine, elle vit que les mini-ordinateurs étaient en train de sonner la fin de la courte épopée du "computer utility" et elle chercha à bifurquer vers ce nouveau débouché à fort potentiel.
Elle avait des relations chez HP et redéveloppa ManMan pour tourner sur un mini de cette société. Cette nouvelle version de son progiciel était destiné à être vendue directement par la force commerciale de HP, contournant ainsi encore une fois le problème de la barrière commerciale (en 1976, époque où HP débuta la commercialisation de ManMan, ASK était encore une toute petite entreprise). Mais Sandra Kurtzig réalisa vite que HP faisait plus de profit qu'ASK sur ces ventes et, une fois que sa société en fut capable, elle décida de bifurquer encore une fois : en passant à la commercialisation directe de son produit.
Cependant, elle ne se contenta pas de vendre ManMan tel quel, elle eu l'idée de le vendre intégré à des systèmes complets (toujours sur des minis HP). C'est ainsi qu'ASK pris son envol comme intégrateur plutôt que comme simple éditeur de logiciels (alors que son activité principale était tout de même focalisée sur le développement et l'évolution de ManMan).
La suite fut moins glorieuse pour ASK qui perdit un peu son chemin emporté par sa croissance : en 1990, ASK racheta Ingres (l'éditeur du SGBDR éponyme) avec un montage financier compliqué. Ce rachat marqua le début de la fin pour ASK mais son rôle dans le développement du progiciel pour mini-ordinateurs méritait d'être souligné. Le succès de ManMan en tant que MRP de référence dans le secteur industriel annonçait aussi la généralisation des ERP dans tous les autres secteurs (ce qui arriva dans les décennies qui suivirent).

Si la diffusion des MRP était le premier maillon de ce qui allait permettre aux éditeurs de dominer le marché, la montée des SGBDR fut certainement le second...

La montée des SGBDR

Alors que le marché des bases de données semblait clairement réparti dans les années 80 avec les SGDB relationnels pour le décisionnel et les SGBD traditionnels pour le transationnel, cette frontière s'effaça dans les années 90 et les SGDR se généralisèrent. Mais, avant cela, le chemin fut long et difficile. Nombreux doutaient que les SGBDR deviennent plus qu'une curiosité de laboratoire pour universitaires.

Charles Bachman lance le mouvement

La préhistoire des bases de données repose presque entièrement sur un nom : Charles Bachman. C'est lui qui, en 1961 et chez General Electric, développa le premier IDS (pour Integrated Data Store), un système pionnier de base de données capable de prendre en compte les nouveaux supports de stockage comme les disques magnétiques qui commençaient à apparaitre.

Les tous premiers logiciels de bases de données datent donc des années 60. La technologie d'alors était encore assez frustre et les principes retenus ne permettaient pas l'indépendance des applications avec le système de base de données utilisé.
C'est aussi Bachman qui au milieu des années 60 rassembla le DBTG (Database Task Group) qui allait publier une série d'articles et de spécifications sur les méthodes d'accès aux bases de données par des langages comme Cobol.
Le DBTG fut intégré à CODASYL (déjà responsable de la normalisation du Cobol) et produisit une norme qui fut adoptée aussitôt par de nombreux constructeurs… Sauf IBM !
C'est qu'IBM, de son côté, avait lançé sur son marché son propre SGBD (IMS, en 1968) qui était dérivé d'un projet réalisé pour la NASA pendant le programme Apollo. Jusqu'à l'apparition du modèle relationnel, les principaux systèmes étaient soit hiérarchiques (comme IMS d'IBM ou Total de Cincom)) soit navigationnels (ou réseau, conformes au modèle proposé par le DBTG issu de CODASYL en octobre 1969) comme IDMS (Computer Associates), Datacom DB (d'ADR, plus tard rachété par Computer Associates), ou Adabas (Software AG, Adabas -mais aussi Datacom DB- a été largement modifié avec le temps et a même été adapté au modèle relationnel…).
C'est ce modèle navigationnel promu par CODASYL qui provoqua une poussée de développement dans le monde des bases de données jusqu'alors pas très dynamique. Quand General Electric décida d'abandonner le secteur informatique, c'est la SSII de John Cullinane qui racheta leur base de données et s'occupa de la migration du logiciel pour la porter sur mainframe IBM (mais il eu aussi des versions pour DEC et ICL). IDMS (c'est le nom que donna Cullinane à ce produit) était conforme à la "norme" CODASYL alors que les autres bases de données qui dominaient jusqu'alors le marché (IMS et TOTAL) ne l'étaient pas… Du coup, IDMS connut une croissance foudroyante et ses ventes passèrent de $2 millions en 1975 à $20 millions en 1980.
Le développement d'applications sur ces SGBD n'était pas facile et c'est pour cela que certains éditeurs proposaient d'associer leur base de données avec un langage de programmation de haut niveau qu'on a vite baptisé L4G (langage de 4ème génération). Les deux "couples" (SGBD + L4G du même éditeur) les plus fameux du début des années 70 étaient sans doute Datacom/Ideal (d'ADR) et Adabas/Natural de Software AG. Adabas (qui est une abréviation de "adaptable database system") vient de la société allemande Software AG. Adabas a été développé par Software AG avec l'aide de l'AVI Institut. Cet éditeur était le seul acteur européen un peu important (avant que SAP le dépasse et le fasse un peu oublier) dans un secteur entièrement dominé par les sociétés américaines.
On reviendra sur cette épisode des L4G car il le mérite et revenons à nos acteurs… Charles Bachman reçu le prix Turing en 1973 et, à cette occasion, prononça le fameux discours "The programmer as navigator" (le programmeur comme navigateur). Charles Bachman est alors à son apogée mais intéressons-nous plutôt à celui qui va lui succèder… Un certain Codd.

Codd pose les bases du relationnel
C'est Edgar F. Codd qui en 1970 posa les bases du modèles relationnel. Dans une série d'articles (Edgar F. Codd rédige en premier l'article fondateur "A Relational Model of Data for Large Shared Data Banks" dans la revue Communications of the ACM -Association for Computing Machinery), Codd proposa un modèle qui reposait sur deux principes : 1) l'indépendance des données par rapport à leur mode de stockage matériel, 2) une navigation automatique grâce à un langage de requêtes de haut niveau qui évite au programmeur de devoir "naviguer" d'un enregistrement à l'autre pour trouver la donnée qu'il cherche...
Associé à un langage d'interrogation "ad hoc", les bases de données relationnelles représentaient effectivement une vraie perçée par rapport à la complexité et à la rigidité des systèmes de bases de données en usage au début des années 70. Avec une base de données classiques (conforme ou non au modèle CODASYL), vous ne pouviez pas obtenir le résultat d'une requête sur votre écran simplement en la tapant depuis le clavier de votre terminal connecté au système central de votre organisation… Non, il fallait passer par un programmeur qui allait coder votre requête dans un programme qui, une fois mis

au point, sera exécuté en batch et le résultat vous arrivera sur un listing… Avec un processus aussi lourd, on imagine facilement la frustration de l'utilisateur en s'apercevant que la requête était incomplète ou pire, erronée !
Il fallait recommencer tout le parcours en espérant avoir un accès rapide au programmeur capable de s'y retrouver dans la structure de la base de données… Le relationnel permettait de résoudre tout cela en une seule fois : on avait enfin une structure de base de données à peu près lisible, un langage d'interrogation assez facile à manipuler sinon à apprendre (en tout cas, bien moins complexe et rébarbatif qu'un langage de programmation, même le plus basique…) et, cerise sur le gateau, un logiciel permettant de saisir la requête au clavier et d'avoir le résultat immédiatement (enfin, plus ou moins rapidement selon la complexité de la requête, la taille de la base de données et la qualité de son indexation…) à l'écran, un vrai paradis par rapport à la boucle programme-batch-listing !
Donc, Codd proposait donc cela à travers sa série d'article publiés en 1970 mais rien d'autre… Il fallu attendre un peu avant de voir les premières implémentations expérimentales.

System/R chez IBM, l'enfant non désiré !
Codd travaillait chez IBM mais big blue n'avait pas l'intention de remettre en cause IMS qui fonctionnait bien pour une idée dont on ne savait pas encore si elle était vraiment réalisable (et si oui, amenait-elle effectivement les bénéfices espérés ?). Mais Codd sut mener une action de lobbying intelligente, y compris en acceptant un débat avec Charles Bachman qui représentait alors l'autorité suprême en matière de bases de données… Tant et si bien qu'IBM fut quasiment "forcé" de lancer un projet de recherche au laboratoire de San José pour vérifier la faisabilité des travaux de Codd.
Le projet s'appelait System/R et la première phase (en 1974/75) permit de produire rapidement un prototype qui démontrait la validité des principes énoncés par Codd. Le code de ce premier proto ne fut pas repris par la suite car il s'agissait vraiment d'un premier jet. La seconde phase (en 1978/79) produisit une version plus complète fonctionnellement, permettant l'accès concurrentiel (multi-utilisateurs) et surtout, dotée d'un langage d'interrogation qui fit du chemin depuis… SQL.
System/R était un projet de recherche et il ne fut donc jamais commercialisé en tant que tel. On sait que le premier SGDR d'IBM fut SQL/DS proposé en 1982 pour les mainframes tournant sous VM et VSE. Pour la version MVS, il fallu attendre 1983 et le fameux DB2. Ce qu'on sait moins c'est que le code de System/R fut tout de même réutilisé en partie pour la toute première implémentation d'un SGBDR chez IBM : celui qui était intégré au système d'exploitation du System/38 en 1980 (le 38 était le mini-ordinateur de Big Blue, successeur du 36 et avant l'AS/400). Cette implémentation était encore une retombée du projet "Future Systems" d'IBM, qui prévoyait de gérer des BDD relationnelles en mode natif... L'équipe FS avait fait l'inventaire de tous les développments "pirates" à l'intérieur d'IBM (VM, APL, System/R, etc.) dans le but de les intégrer.
A la suite de System/R, Larry Ellison créa Oracle en 1977 (mais, dans un premier temps, la société s'appelait Software Development Laboratories, puis en 1979 le nom changea pour Relational Software, Inc. -RSI- et, enfin, en 1982, RSI changea encore pour Oracle Systems).
Mais avant Oracle, Micheal Stonebraker, chercheur à Berkeley, développa un prototype universitaire de SGBDR à peu près à la même époque qu'IBM avec System/R et lui aussi en s'inspirant de l'article de Codd.

Ingres et la vertu de l'essaimage
Codd posa les bases mais c'est bien Stonebraker qui sema la première graine !
Alors que Codd se débattait pour qu'IBM fasse enfin quelque chose avec ses idées, Micheal Stonebraker agissait sans perdre de temps. Aidé d'Eugen Wong, il trouva un peu d'argent afin de lancer un projet de recherche basé sur le modèle relationnel et ce dès 1973. Initialement, les fonds sont venus de la branche études économiques de Berkeley afin de réaliser un système de modélisation géographique. C'est pour cela que le projet

s'appelait Ingres (qui voulait dire Interactive Graphics and Retrieval System et non une référence au célèbre peintre français du XIXème siècle…).
Au bout du compte, c'est surtout la NSF (National Science Foundation) qui finança le projet Ingres qui dura jusqu'en 1982 (puis fut suivit de Postgres destiné à promouvoir les bases de données objets mais c'est une autre histoire et qui ne connut pas un dénouement aussi heureux que pour le relationnel !). Ingres était très semblable à System/R mais reposait sur des Minis de DEC plutôt que sur un maiframe IBM. Une première version était "montrable" dès 1974 mais fut suivie par plusieurs révisions afin de rendre le code plus maintenable. Très vite, Stonebraker commença à diffuser le code d'Ingres en dehors de Berkeley afin de reccueillir des commentaires et des idées sur le prototype qui évoluait en permanence.
En 1976, Ingres fut doté d'un langage de requêtes (QUEL) très semblable à SQL (semblable mais différent tout de même…). On sait que dans l'affrontement entre QUEL et SQL, c'est ce dernier qui triompha mais le riche héritage d'Ingres va bien au-delà de cette anecdote : durant les années "Berkeley" du projet Ingres, pas moins de 30 personnes ont travaillé dessus à tour de rôle (mais il n'y eu jamais plus de 6 programmeurs à la fois sur le code en évolution permanente). Si on ajoute les 15 programmeurs qui ont développé et corrigé le code de System/R, ça nous fait une petite cinquantaine d'individus qui se sont ensuite éparpillés autour de Berkeley et de San José pour créer effectivement les éditeurs qui ont donné force et vie au modèle relationnel. En effet, tous ceux qu'on retrouvent impliqués dans les multiples projets des années 80 autour des SGDR sont passés à un moment ou à un autre soit à Berkeley sur Ingres, soit à San José sur System/R.
Michael Stonebraker lui-même créa Ingres Inc pour commercialiser le code de son projet. En 1980, Roger Sippl et Laura King fondent Relational Database Systems (RDS et qui deviendra Informix en 1981) et créent eux aussi un SGBDR basé sur Ingres.
Bob Epstein participa à la fondation de Britton-Lee avant de rejoindre Sybase et ainsi de suite. Epstein explique bien que "ce qui resta d'Ingres est l'expérience d'avoir un prototype et de savoir quelle parties devaient être développées d'une façon différente…" et pour une technologie encore immature comme l'était le relationnel à l'époque, c'était simplement énorme.

Oracle, l'enfant prodige
En 1977, Larry Ellison s'associe à Robert Miner pour créé une société de service (appelée d'abord System Development Laboratories, elle deviendra Relational Systems Inc en 1978 puis enfin Oracle Systems en 1982) dont le premier contrat est de développer un système de base de données pour la CIA. Mais Ellison a fait partie de l'équipe Amdahl qui a créé le premier mainframe compatible IBM et, de cette expérience, il a gardé l'habitude de surveiller ce que prépare Big Blue. Et, justement, il entend parler de l'intense activité autour de la notion de base de données relationnelle que prépare IBM.
La petite société d'Ellison et Miner a commencé son activité à Belmont, dans le nord de la Californie. Ses premiers concurrents venaient également de la même région, tous voisin du laboratoire IBM de San José où avaient été menées les recherches sur System/R. Larry Ellison embaucha des programmeurs qui avait participé à Ingres et il developpa un SGBDR doté du langage SQL avant même qu'IBM dévoile SQL/DS. Il avait eu vent du langage SQL à travers les articles publiés par le groupe qui travaillait sur System/R.
Dès 1978, la petite société annonce qu'elle développe un SGBDR qui sera le premier a intégrer le langage SQL que prépare IBM. Ce premier "produit" sera effectivement disponible en 1979 (c'est Oracle version 2 car il n'y a jamais eu de version 1… cette version est proposée pour $48000 pour les minis de DEC).
Une version corrigée sort en 1981 portée sur les minis de DEC, PDP11 et Vax. C'est également à partir de cette version qu'apparait la toute première version Forms, l'environnement de développement associé à Oracle. La portabilité est le mot-clé des premières version du SGBDR Oracle et dès 84, VM d'IBM vient s'ajouter à VMS de DEC en plus de UNix et ses nombreuses variantes. Durant les années 80 et 90, les différentes versions d'Oracle de la V4 à la V6 auront été portées sur une multitude de minis-ordinateurs et de stations de travail, tant sur différentes versions d'Unix que sur des systèmes d'exploitation propriétaires de ces constructeurs de Minis.

Ces portages étaient souvent sous-traités et il n'était pas rare de devoir retrouver LE programmeur qui avait été chargé du portage afn de corriger le bug qu'un client découvrait sur site à l'occasion d'une utilisation inédite (comprendre "qui n'a pas été testée…"). Je me souviens que Thomson CSF TVT utilisait des minis de la marque "Encore" dans les années 80 diffusés par Matra en Europe s'était retrouvé avec un bug bloquant sur la version Oracle qui tournait sur ces machines… Le support d'Oracle France fait appel à celui de la maison mère pour trouver la solution… Peine perdue, il faut retrouver les auteurs du portage qui, comme d'habitude, a été sous-traité. Le prestataire est une SSII canadienne mais celle-ci avoue qu'elle l'a elle-même sous-traité à un developpeur indépendant que le support finit par localiser au fin fond de l'Alaska !
Une fois le développeur ermite extrait de son refuge, on lui soumet le bug, il trouve la correction, l'applique et repart dans son isolement… C'était les temps héroïques des débuts du relationnel…
Ellison et consorts ne s'arrêtaient pas à ce genre de détails : la technique progressait vite et les bugs "faisaient partie du jeu". En 1984, un fond de capital-risque s'intéressa à Oracle et cet investissement lui permit de poursuivre une croissance rapide.
En 1985 apparait la version 5 d'Oracle et avec elle SQL*Net, le middleware qui permet le fonctionnement client-serveur (un PC sous MS-Dos devient capable d'interroger une base Oracle qui tourne sous Vax ou Unix).
Toujours plus fort, en 1986 c'est SQL*Star qui est dévoilé : ce middleware va encore plus loin que SQL*Net et permet à plusieurs versions d'Oracle server de fonctionner de façon distribuée. A la même époque, Ingres et Informix proposaient plus ou moins le même type de réalisations spectaculaires mais qui étaient plus des démonstrations dans le cadre d'une guerre commerciale où la surenchère était la règle que de fonctionnement pleinement opérationnel et utilisable sur sites.
En 1986, c'est également l'année de l'entrée en bourse pour la société dont le chiffre d'affaires atteignait alors $50 millions avec une base installée de 3000 mini-ordinateurs et mainframe faisant tourner son SGDR phare.

En vérité, il faut attendre 1986 pour voir la version 6 d'Oracle qui propose enfin un fonctionnement possible en transactionnel avec un verrouillage au niveau ligne et 1992 avec la version 7, première version vraiment achevée du SGBDR qui intégre à son tour - et progressivement- les fonctions avancées telles que déclencheurs et procédures stockées "inventées" par Sybase.
Mais c'est du côté du dynamisme commercial qu'Oracle aura vraiment étouffé ses rivaux : de 1979 à 1990, les ventes du SGBDR auront progressé d'au moins 100% chaque année !
La politique commerciale d'Oracle était très agressive avec des commissions parfois énormes pour ses meilleurs vendeurs. Cette expansion forcenée ne se fit pas sans quelques incidents de parcours : en 1991, Oracle essuya une grosse crise qui faillit l'emporter. Mais Larry Ellison repris les choses en main et put remettre sa société sur de bons rails.
On l'a vu, les tous premiers SGBDR disponibles comme Oracle souffraient d'un grave défaut : ils étaient terriblement gourmands en ressources et aussi très lents lors de l'exécution de requêtes complexes (et la lisibilité de SQL ou de Quel favorisait l'écriture de requêtes complexes, forcément…).

La parenthèse des "machines base de données"
Pour faire face à cette consommation de ressources qui effrayait les responsables de sites informatiques et les faisait bannir d'avance l'idée même d'installer un SGBDR (fut-il doté de l'étiquette magique IBM…) sur leurs précieux mainframes, certains entrepreneurs eurent l'idée de réaliser des serveurs dédiés et optimisés pour l'exécution de ces gourmands SGBDR… Autrement, dit des machines base de données !
La notion de machine dédiée faisait partie des idées qui faisaient fureur à l'époque, tout comme les stations de travail dédiées dédiées au langage LISP. Mais cette notion fit long feu et seul Teradata dans le domaine des bases de données put faire fructifier cette idée pendant quelques années.

Justement, Teradata faisait partie des pionniers de cette vague de serveurs dédiés, l'autre étant Britton-Lee (où on retrouve Bob Epstein parmi les fondateurs). Pour comprendre pourquoi ce concept connut quelques succès, il suffit de se replacer dans le contexte de l'époque et, pour cela, donnons la parole à un témoin, Paul Stuber, responsable de l'analyse des ventes de la société Wrigley (les chewing gums)… Stuber est l'heureux client d'un serveur Britton-Lee et il explique en 1988 ce que ça lui apporte :
"il s'intégre bien avec nos PCs et c'est comme un appareil spécialisé pour nous. Il n'est pas affecté par ce qui se passe sur notre mainframe et il ne le perturbe pas non plus. De plus, la sécurité est plus facile à assurer sur cette simple machine.
Les SGBDR font du bon travail mais ils exigent une très grosse machine. Celle-là est petite et notre service suffit pour la gérer. Nous pouvons ainsi offrir des données en ligne à nos forces de vente et marketing sur n'importe quel PC ici."
Le témoignage de Stuber est éclairant sur son avantage principal : le serveur de données dédié permet d'avoir un serveur d'infocentre indépendant du mainframe. Il suffisait pour cela de mettre en place la procédure et le traitement permettant de "déverser" les données utiles (stockées dans un SGBD traditionnel le plus souvent) depuis le mainframe vers la machine base de données.
Mais alors, si ce concept était si pratique, pourquoi Britton-Lee n'a-t-il pas prospéré (et Teradata est de son côté resté un fournisseur cantonné dans une niche : les très grosses bases de données) ?
Tout d'abord parce que ces solutions étaient coûteuses (voire même très coûteuses dans le cas de Teradata) et que leur avantage initial en matière de performances n'est pas resté valide longtemps face aux progrès constants des serveurs Unix issus du secteur des stations de travail (ceux-ci évoluant sans cesse et très vite… des machines conçues et fabriquées sur mesure en petites quantités ne pouvaient pas suivre face à des machines polyvalentes produite en masse).
Ensuite parce que les logiciels de SGBDR ont progressé. Donc, au tournant des années des années 90, un serveur Unix équipé d'Oracle, Informix et Ingres présentait à peu près le même potentiel de performances qu'un serveur Britton-Lee avec, en plus, l'avantage de la polyvalence (le serveur Unix pouvait être recyclé pour un autre type d'application, réutilisation impossible avec une machine base de données), le tout pour moins cher.

Maturité et lutte commerciale
Un début de maturité du marché commence à voir le jour au début des années 90 et, en dehors du monde fermé d'IBM, la lutte commerciale se résume à un affrontement entre Oracle, Informix et Ingres (plus quelques autes éditeurs de moindre importance et qu'on a oublié depuis…). Chacun de ses trois acteurs se dispute sur le thème "c'est moi qui suis le plus relationnel de tous !" et chacun se contente de jouer sur le terrain du décisionnel (les SGBDR sont encore considérés comme bien trop lents pour être utilisés pour des applications de production).
Sybase fut le premier éditeur de SGDR à aller jouer sur le terrain du transactionnel. L'idée de Mark Hoffman, Bob Epstein, les fondateurs de la société, était de faire l'équivalent logiciel d'une machine dédiée aux bases de données mais sans ses inconvénients. Le résultat fut SQL Server qui ouvrit une nouvelle voie pour les SGBDR. SQL Server est véritablement le premier SGBDR moderne : il est doté d'innovations techniques internes qui font la différence comme les procédures stockées et les déclencheurs. Il est aussi fourni avec des librairies dynamiques qui permettent le développement d'applications sans avoir à passer par un pré-compilateur.
Cette modernité a séduit Bill Gates qui passe un accord avec Sybase pour porter SQL Server sur OS/2 (et sur Windows NT par la suite, l'accord permet aussi à Microsoft d'utiliser le nom "SQL Server" et la firme de Redmond va en faire un tel usage que Sybase préférera renommer son produit par la suite pour éviter les confusions…).
Sybase ne fut pas seule à s'engoufrer dans la brêche et c'est Oracle qui en consolida sa position en devenant N°1 du secteur dès le début des années 90.

Informix su trouver une niche en s'embarquant au sein de nombreux progiciels destinés à des petites machines Unix mais Ingres fut la principale victime de cette lutte commerciale, pourquoi ?
Le rachat infructueux par ASK n'aida certainement pas à ce que le produit pionnier trouve son marché mais il semble que ce soit surtout par ambition qu'Ingres se soit perdu en chemin : pour retrouver un avantage en performance face à Oracle et Sybase, Ingres inc consacra beaucoup de ses ressources à un projet mené conjointement avec Sequent (un constructeur de minis spécialiste des traitements faisant appel à de multiples processeurs fonctionnant en parallèle). Le projet avec Sequent consomma de l'argent mais fut long à déboucher. Finalement, Ingres fut obligé d'arrêter les frais et c'est Informix qui récupéra la mise en remplaçant Ingres au pied levé auprès de Sequent !
Le rachat par Computer Associates fut la touche finale dans ce long processus de mise à mort… Contrairement à ses habitudes, le management de CA voulait garder les programmeurs d'Ingres car ils avaient été duement prévenus que des spécialistes du noyau d'un SGBDR ne courraient pas les rues… Mais le management très "côte est" de CA s'accomodait mal de la décontraction et des moeurs de l'équipe de développement très "cote ouest" d'Ingres et CA commit l'erreur fatale : exiger des programmeurs qu'ils signent une clause de non-concurrence s'ils voulaient que leur contrat soit transféré sous la bannière CA… Une précaution qui semblait logique dans le cadre d'une gestion rigoureuse mais qui provoqua la fuite de l'équipe qui était alors courtisée par tous les concurrents (on raconte même qu'Oracle faisait tourner des mini-bus aux couleurs d'Oracle autour de l'immeuble d'Ingres pour proposer des contrats aux sortants !). Sans équipe de développement pour le maintenir et le faire évoluer, Ingres ne survit pas longtemps au sein du catalogue de Computer Associates… Un rachat pour rien ou presque !

Informix racheté par IBM en avril 2001, Sybase un peu effacé, c'est désormais Microsoft qui reste pour donner la réplique à Oracle, solide leader de ce secteur désormais bien mature (nous évoquerons l'épisode MySQL dans la section consacrée à l'Open Source).

Il est juste d'ajouter que les SGBDR n'ont pas triomphé seulement grâce à leur éventuelle supériorité technique mais aussi parce qu'ils ont pu croitre sur un marché que les éditeurs de SGBD traditionnels leur ont laissé libre jusqu'à ce qu'il soit trop tard : celui des stations de travail sous Unix.
Pourquoi une telle erreur tactique ?
Déjà, les éditeurs de SGBD traditionnels préféraient s'affronter sur le terrain des mainframes plutôt que sur celui des mini-ordinateurs pour deux raisons : 1) le prix des licences sur mainframe était plus élevé que sur mini-ordinateur (alors que l'effort de vente était quasiment le même) 2) les mini-ordinateurs étaient plus destinés à faire tourner des progiciels que de servir de plate-formes pour des applications spécifiques "développées à la maison" (in-house programming), terrain de prédilection des SGBD sur grands systèmes.
Donc, le marché des stations de travail apparaissait comme un sous-ensemble du marché des minis déjà considéré comme peu attractif pour ces éditeurs. De plus, leur principal argument résidait dans la stabilité technique et la fiabilité de leur SGBD, un argument peu séduisant pour le monde des stations de travail en perpétuelle ébullition et où seule la performance et l'innovation étaient considérées.
Au contraire, ce contexte allait comme un gant aux éditeurs de SGBDR qui s'accommodaient bien d'une instabilité technique propre au renouvellement constant des stations de travail puisque cela permettait de masquer leur propre manque de fiabilité. Et le côté "technique de pointe" du relationnel (qui était également accompagné de mises en oeuvre avancées comme le mode client-serveur ou le fonctionnement distribué) était un argument fort pour les clients des stations de travail, réputés pour privilégier la performance et l'innovation dans leurs choix.
Au début des années 80, la technologie relationnelle apparaissait (avec quelques raisons) comme trop immature pour intéresser les fournisseurs de SGBD traditionnels (sauf IBM) permettant à Oracle et consorts de se développer au sein de leur "niche"…

Seul IBM a su suivre le mouvement et c'est en 1985 que Cullinet annonça son intention de rendre IDMS complétement conforme au modèle relationnel… Cette annonce ne servit qu'à rendre légitime et crédible ce nouveau modèle et les cients de Cullinet se sont alors tournés vers Oracle et surtout vers IBM pour "commencer à y goûter".
Bref, pour toutes ces raisons, les "anciens" acteurs de ce marché comme Cullinet ou ADR étaient poussés vers la sortie (le plus souvent rachetés par CA justement !) et seul IBM put rester un acteur significatif de ce marché avec son offre traditionnelle (IMS) allant de pair avec son offre relationnelle (DB2).

A peu près au même moment que la montée des SGBDR, on assista à une autre évolution majeure et complémentaire du logiciel pour entreprise, les L4Gs...

Les L4G et le problème du développement d'applications
Pour faire face au problème du goulot d'étranglement du développement d'applications, certains éditeurs eurent l'idée de faire évoluer les langages de développement vers un plus haut niveau d'abstraction. Cette évolution fut vite appelée 4ème génération et ces langage des L4G. Les L4G étaient le souvent accompagnés de leur propre SGBD ou, pour les éditeurs ayant déjà un SGBD Codasyl ou non à son catalogue, fortement couplé au SGBD de ce dernier.

Phase 1, les L4G pour mainframes en 70' & 80'
Les premiers L4G sont d'abord apparus sur les mainframes avec des produits comme Mapper, Ramis, Nomad, Focus, SAS, Ideal et Natural. RAMIS (pour Rapid Access Management Information System), disponible en 1970, est souvent considéré comme le premier d'entre eux. Ces L4G visaient principalement l'environnement des grands systèmes IBM avec quelques exceptions comme Mapper (qui voulait dire MAintain, Prepare, and Produce Executive Reports) qui avait développé par Sperry Univac pour ses propres mainframes…
Ramis était disponible via le système de temps partagé proposé par National CSS (avant d'être racheté par Computer Associates, comme d'habitude) mais aussi sous forme de logiciel à installer. Gerald Cohen qui avait développé Ramis avec Peter Mittleman quitta Mathematica (l'éditeur de Ramis) pour fonder Information Builders (IBI) et redevelopper Ramis sous le forme de Focus qui commença à être commercialisé en 1975. Pendant ce temps, NCSS décida en 1973 de travailler sur son propre produit (Nomad pour NCSS Owned, Maintained, And Developed, disponible en octobre 1975) plutôt que dépendre de Ramis produit de Mathematica. C'est cette origine presque commune qui explique pourquoi les trois produits (Ramis, Focus et Nomad) étaient très ressemblants, jusque dans leurs syntaxe quasiment identique.
Un exemple de la puissance des L4G par rapport au Cobol est illustré par la solution à une question classique connue sous le nom du "problème de l'ingénieur" : donner une augmentation de 6% aux ingénieurs dont la note moyenne est de 7 ou mieux. Pour résoudre ce problème, James Martin rédigea une douzaine de pages de code Cobol et seulement deux pages de code Mark IV. Avec Nomad, ce problème pouvait être résumé en une simple ligne : CHANGE ALL SALARY=SALARY*1.06 WHERE POSITION='ENG' AND AVG(INSTANCE(RATING)) GE 7…

Mais, bien entendu, cette simplicité se payait via une consommation de ressources importante : les L4G étaient bien plus lents en exécution que les programmes rédigés avec des langages traditionnels. A une époque, on a pu croire que la simplicité et la lisibilité des L4G permettrait de se passer de programmeurs… Que les utilisateurs allaient eux-mêmes rédiger les propres programmes !
Ce fantasme s'est répété plusieurs fois depuis et il est toujours resté ce qu'il était au début : un fantasme. On a remis cela au début des années 90 avec l'arrivée de la programmation objet… Il était fréquent d'entendre des déclarations comme "Demain on n'aura plus besoin de développeurs. Un chef de projet n'assemblera que des briques". Ce type de prédictions -erronées- a été fréquent tout au long de l'histoire de l'informatique : COBOL était verbeux car on s'imaginait, qu'ainsi, les utilisateurs pourraient coder eux-

même si le langage de programmation présentait un vocabulaire similaire à celui du langage courant, au
début du SQL beaucoup de gens ont prédit que l'utilisateur final allait écrire lui-même ses requêtes SQL du style SELECT FROM WHERE (j'ai même assuré une formation au SQL en 1988 pour des responsables des ventes d'un grand distributeur…) et la même méprise s'est répétée avec la méthode Merise où de nombreux utilisateurs ont été formés en vue de rédiger eux-mêmes les diagrammes de conception...

Cependant, les L4G représentaient bien une réponse appropriée à un besoin qui était latent lors des années 70 et 80 (surtout dans les environnements mainframes) : le goulot d'étranglement des programmes était surtout dù au fait qu'il fallait passer par le service informatique et ses programmeurs pour la moindre intervention sur les applications. Si un cadre avait besoin d'un nouveau rapport sur les ventes, il fallait écrire une nouvelle application pour cela et ce "simple" rapport se transformait en projet qui durait des mois (sans compter qu'il fallait des mois sinon des années avant que cette demande soit prise en compte…).
C'est que les applications de production (celles qui nécessitaient des saisies intensives et des temps de réponses très courts) avaient été développées lors des décennies 60 et 70. A la charnière entre les années 70 et 80, elles étaient désormais opérationnelles et stabilisées. Les demandes des utilisateurs se reportaient donc naturellement vers des besoins complémentaires et c'est ainsi qu'on est passé progressivement à la notion du décisionnel.
Les L4G étaient particulièrement bien taillés pour ces applications d'aide à la décision puisqu'ils étaient tous plus ou moins orientés vers la production de rapports basés sur des extractions issues des systèmes de bases de données. Le côté analyse statistique est bien illustré par un autre produit vedette de cette période : SAS.
Fondée en 1976, SAS Institute Inc proposait initialement un ensemble de logiciels destiné à l'analyse statistique pour mainframe IBM. En plus du classique fonctionnement batch, SAS présentait un fonctionnement interactif innovant : un environnement découpé en plusieurs cadres (on ne disait pas encore fenêtres à cette époque) où le programme pouvait être édité dans un premier cadre, le résultat pas à pas dans un second et le rapport final dans un troisième. SAS évolua pour devenir un L4G à part entière et connu un succès particulier dans le domaine pharmaceutique où les applications baties avec ces fonctions évoluées permettaient de passer les examens de mise sur le marché exigés par la FDA (Food and Drug Administration, l'équivalent américain des ministères de la santé et de l'agriculture).
Les premiers éditeurs de SGBDR ont également développé leurs propres L4G (Forms pour Oracle et 4GL pour Informix) afin de faciliter les développements sur leurs bases de données. Mais ces produits n'étaient pas prévus pour adresser d'autres cibles et cette fermeture a été forcément un facteur limitant dans leurs diffusion.
Donc, les L4G ont été les bienvenus pour tout ce qui était du domaine de l'infocentre. En revanche, ils ont moins bien réussi dans le domaine du transactionnel (sauf les L4G qui était des compagnons de bases de données classiques comme Natural avec Adabas ou Ideal avec Datacom). C'est d'ailleurs une constante de l'histoire de l'informatique : les nouveaux entrants ont intérêt à se trouver une niche vierge plutôt de d'attaquer de front les positions établies. Les mini-ordinateurs se sont installés tranquillement parce qu'ils ont inaugurer une première forme de décentralisation appelée "informatique départementale" qui venait en complément des systèmes centraux (dans les grandes organisations, dans les PME, les minis étaient souvent les premiers et seuls ordinateurs) et non en remplacement. Et il en a été ainsi de la plupart des nouvelles vagues qui sont venues s'ajouter à l'existant (avec plus ou moins de recouvrement) sans prétendre à tout remettre en cause (mêmes si les promoteurs les plus zélés annonçaient une révolution en cours qui allait tout bouleverser, la plupart du temps, il n'en a rien été).
C'est parce qu'ils s'étaient trouvé un nouveau marché (les applications décisionnelles) que les L4G ont pu croitre et prospérer jusqu'à un certain point. Mais, alors que la mini-informatique n'avait rien changé au rapport des forces (certains des L4G issus des grands systèmes ont été portés sur Vax et HP3000 ainsi que sur des systèmes Unix, d'autres sont nés directement sur des minis comme Progress par exemple), c'est la

montée des PC et surtout des interfaces graphiques qui va renouveller le marché en profondeur...

Phase 2, les environnements RAD pour le client-serveur

Alors que les environnements de développement client-serveur sous Windows ont monopolisé l'attention dans les années 90, le tout premier L4G graphique est apparu bien avant mais pas sur Windows. En effet, c'est sur Macintosh qu'eu lieu la perçée majeure (mais peu remarquée à l'époque) avec 4ème Dimension commercialisé par l'éditeur français ACI.

En fait, le logiciel de développement pour Mac s'appelait "Silver Surfer" pendant que Laurent Ribardière en écrivait le code. Il était prévu qu'Apple le distribue directement sous ce nom mais, sous la pression des autres éditeurs qui jugeaient que cela représentait un désavantage majeur pour eux, Steve Jobs renonça à ce projet pour se concilier les éditeurs. Finalement, Ribardière hérita de son logiciel et pu lui trouver un éditeur avec ACI.

4ème Dimension était un environnement formidable pour son époque : facile à utiliser, performant et très fidèle à l'interface du Mac (même si les premières versions étaient bien buggées… Tout comme les premières versions du Mac d'ailleurs !). Alors que 4D était disponible depuis 1985, il fallu attendre encore 4 ans avant qu'un environnement graphique plus ou moins équivalent apparaisse sur Windows avec SQL Windows proposé par Gupta.

Uman Gupta était un proche collaborateur de Larry Ellison à Oracle quand il a quitté l'éditeur suite à un désaccord stratégique : Gupta ne voulait pas se contenter de porter la version 5 d'Oracle sur PC mais plutôt redévelopper une version spécifique. Ellison n'était pas de cet avis et voulait se contenter d'un portage pour un marché où, selon ses dires "il n'y avait pas d'argent à gagner"… Gupta voyait au contraire un avenir radieux sur le PC et en profita pour lancer sa propre société. Son premier produit fut un SGBDR spécifiquement développé pour le PC : SQL Base, un SGBDR conçu dès le départ pour fonctionner en serveur avec des applications clientes distantes. Mais le produit qui distinguait vraiment Gupta de ses compétiteurs était son L4G qui était le tout premier environnement de ce genre sous Windows et nativement client-serveur (sous Windows 2.1 en plus !).

Pendant 2/3 ans, SQL Windows fut quasiment le seul produit disponible sur ce marché neuf mais très en vue dans les médias spécialisés qui commençaien-t à évoquer le client-serveur comme l'état de l'art à adopter séance tenante pour moderniser l'informatique d'entreprise qui, effectivement, commençait à dater un peu. Si c'est un avantage certain d'être le premier sur son marché, ce n'est pas pour autant une garantie certaine de succès puisque c'est Powerbuilder qui s'imposa rapidement comme l'outil de pointe dans ce domaine.

L'histoire de la naissance de Powerbuilder est assez singulière et mérite en cela d'être contée...

La génése de Powerbuilder

L'histoire de Powerbuilder n'est pas classique du genre "une startup du nom de Powersoft se lance avec un outil développé par quelques programmeurs géniaux associés et fait la conquête du marché en deux temps trois mouvements"… Non, l'histoire de cet outil qui a dominé son époque a débuté au sein d'un acteur bien établi : Cullinet.

L'éditeur du bien connu SGBD IDMS avait racheté au début des années 80 une petite société, Computer Pictures basée à Boston. L'équipe issue de Computer Pictures développa un prototype de logiciel d'accès à IDMS depuis un PC. Ce proto appelé "Golden Gate" avait été développé en 1984 et devait être adapté par la suite aux interfaces graphiques qui commençaient à émerger à cette période dans le sillage du Macintosh d'Apple.

Dave Litwack était le chef de projet de Golden Gate et il voulait que le futur logiciel soit complétement orienté objet car, à cette époque, le C++ était vu comme le successeur du C et Smalltalk était considéré comme le langage de pointe à suivre. Dès 1985, une première version fut montrée à la direction et il était évident que le potentiel de ce projet pouvait être grand.

Mais, dans le même temps, Cullinet faisait face à une période difficile : Computer Associates essayait de racheter la société et les efforts de la direction était mobilisée pour contrer cette OPA hostile.

Pendant ce temps là, une petite société appelée PowerSoft developpait et commercialisait un MPR II (GrowthPower) qui tournait sur HP3000. Mitchell Kertzman, dirigeant de Powersoft réalisa que l'arène du développement pour PC allait connaitre une croissance explosive quand il interrogea ses clients sur l'évolution désirée de GrowthPower… La réponse était claire : tout le monde voulait une interface graphique et un fonctionnement client-serveur !
Il commença alors à chercher un environnement de pointe capable de prendre en compte l'interface graphique. Pour trouver le bon outil, il engagea un consultant indépendant, Dave Litwack (encore lui !). A cette époque (1988), SQL Windows de Gupta était le seul produit sérieux disponible. En dehors de cela, vous deviez coder en C ce que ne voulait surtout pas PowerSoft car ses développeurs habituels n'étaient pas habitués à un langage de bas niveau comme le C.
Dave évoqua son implication dans le prototype développé par Cullinet. Powersoft prit donc contact avec Computer Associates (qui avait finalement racheté Cullinet à ce moment-là) pour explorer la possibilité de racheter le code du prototype en question… CA avait jeté un coup d'oeil sur ce projet pour en conclure qu'il n'avait aucun potentiel (bien vu les gars !). CA était donc content de vendre pour quelque argent un code qui n'avait aucune valeur à ses yeux !

En 1988, 3 ans après les débuts du prototype (commencé à Cullinet), PowerSoft avait le code à finaliser et Dave (désormais employé par PowerSoft) pouvait engager les autres programmeurs qui avaient travaillé sur ce projet. PowerSoft baptisa le nouveau produit "PowerBuilder" et commença à compléter et à améliorer le code. Comme Powersoft était positionné sur les logiciels pour entreprises, son équipe de développement utilisa Powerbuilder pour reprendre les applications existantes (prévues pour les minis VAX de DEC) afin de les adapter à l'environnement client-serveur plus moderne et dans l'air du temps.
Du coup, le "produit" pu être testé dans des conditions d'utilisation réelles et avec des programmeurs "normaux" (pas des top gun capables de faire des applications en C avec le SDK de Windows dans l'état où il était à l'époque…) ayant de vraies applications à développer. Ces tests intensifs furent très utiles pour que la première version de PowerBuilder soit vraiment utilisable.

Pour financer son projet, PowerSoft reçut le support de HP. HP leur donna un "chèque en blanc" après avoir vu une démonstration du produit en cours de finalisation (la beta de PowerBuilder s'appelait "Headstart"). A partir de là, PowerBuilder devint un standard interne au sein de HP. Au siège de Microsoft à Redmond, les développeurs en charge des applications internes faisaient face au même problème que PowerSoft essayait de résoudre : il leur fallait un outil client-serveur permettant la prise en charge complète de l'interface graphique Windows et le tout pour les développeurs d'applications… Ils contactèrent leurs amis chez HP qui leur répondirent que PowerBuilder était le seul outil qui méritait d'être regardé. Début 90, Microsoft devient ainsi le second client de PowerBuilder à l'échelle mondiale. Quelques autres firmes comme American Airlines, 3M, Fidelity Investments et Coca-Cola participèrent au programme de beta pendant les années 90 et 91. La version 1.0 de PowerBuilder a débuté sa commercialisation officielle en juillet 1991. Six mois après ce lancement, Powersoft avait déjà vendu pour $5,2 millions de PowerBuilder !
La version 2.0 suivit moins d'un an après la première version et les ventes montaient toujours ($22 millions en 1992). PowerSoft pu s'introduire en bourse en février 1993 (pour être finalement racheté par Sybase 2 ans après).

D'autres environnements ont quand même réussi à croitre à l'ombre de PowerBuilder. En France, on peut citer NS-DK de Nat Systems.

La gloire de PowerBuilder ne dura pas longtemps : en 1996, la concurrence se réveilla avec le Visual Basic de Microsoft et surtout Delphi de Borland (sorti en 95). De plus, le marché avait encore changé d'orientation : après la vague du client-serveur, vint la ruée sur le Web !

Pourquoi PowerBuilder n'a-t-il pas été capable de prendre en compte le Web ?
En fait, PowerSoft a adapté son produit au développement Web mais pas tout de suite, trop lentement et pas assez en profondeur pour garder son leadership sur ce nouveau marché… Pourquoi cette timidité ?
Parce que l'éditeur n'a pas compris que "moins c'est plus" au moins au début !
Le HTML permettait de simplifier l'interface utilisateur (alors que l'interface graphique était difficilement assimilée par les développeurs ordinaires). Or, c'est justement ce que le marché réclamait à cette époque : un environnement client qui soit plus simple à concevoir (l'interface Windows avec ses multiples widgets représentait un "saut culturel" important pour les développeurs en entreprises habitués au mode caractère…) et qui simplifie le déploiement des applications (et le client Web offrait justement ces deux avantages qui annulait les deux inconvénients majeurs du client-serveur…).
De plus, il est probable qu'il ne s'agisse pas d'une incompétence ou d'un aveuglement de la part des dirigeants de PowerBuilder (qui étaient occupés à digérer la fusion avec Sybase à cette époque clé). Ils ont juste suivi le courant. S'ls avaient demandé aux développeurs de PowerBuilder ce qu'ils voulaient, ces derniers auraient répondu comme d'habitude "plus de fonctionnalités !", et auraient hurlé à l'idée d'une interface (le HTML) où l'on ne peut pas tout gérer au pixel près. Les partenaires PowerBuilder ont sans doute également rechigné devant la simplicité du HTML (car on pouvait croire qu'il y aurait moins d'heures à facturer). Bref, l'écosystème complet a vraisemblablement freiné des 4 fers.
Comme d'habitude, verrouillé dans ses contraintes, appuyé sur ces certitudes et le confort de sa niche d'excellence, l'éditeur dominant n'a pas vu que la nouvelle vague représentait l'avenir... Ou plutôt il l'a compris trop tard : PowerBuilder était modifié timidement pour être capable de prendre en compte la cible Web mais, dans le même temps, un tout autre outil était proposé pour adresser cette cible complètement et en profondeur... Et cet outil était PowerSite, un environnement de développement capable de générer du code (un script nommé "Powerscript", évidemment) aussi bien en direction des serveur Netscape LiveWire que de Microsoft ASP. Mais, avec ce produit, Sybase repartait à égalité avec ses adversaires du moment comme Cold Fusion (et c'est toujours plus difficile de vendre un nouveau produit à sa base installée qui demande plutôt de faire évoluer le logiciel auquel elle est habituée...) et les difficultés de rapprochement entre les équipes Powersoft et Sybase ont sans doute également joué un grand rôle dans la carrière décevante de PowerSite...

======

Encadré : Patrick Smajda, Disctrict Manager chez Powersoft France de 1992 à 1995, évoque les débuts de Powersoft en France...
Je connaissais Samuel Rosenberg du temps ou il travaillait pour Oracle. Par ailleurs j'ai été son premier Client SQL Server sous OS2 (Oupss !) quand il s'occupait de Sybase en France.
A l'époque je dirigeais la SSCI (reOupss !) SRIDE.
2 ans plus tard, Décembre 1991, Je l'ai fortuitement croisé dans les locaux de SQLTech à Boulogne. Il était en négociation avec une petite boite de Burlington (MA), PowerSoft qui avant passée une annonce dans le 01H nationnal (je crois Wired) pour trouver des distributeurs en Europe. Il m'a reçu dans un petit bureau de 9M2 partagé en 2. Première moitié pour son bureau et celui de son assistante Sylvie. Seconde moitié occupée par un stock de 150 boites (format 30x25x20) contenant la doc et les disquettes (re-Oupss !) de la version 1.0 d'un produit dénommé « PowebBuilder » version obsolète 6 mois plus tard..

Il ne m'a pas expliqué comment il s'était fait refourguer ce stock (pratique courante à l'époque) mais je suppose qu'il faillait montrer patte blanche pour négocier le contrat dans de bonnes conditions.
Il m'a demandé mon avis et si j'étais prêt a le rejoindre dans l'aventure comme Simon Azoulay de chez Alten (détail qui aura son importance par la suite) et Sylvie Tuile son assistante. J'étais disponible,
il a prononcé les bons chiffres, j'ai dis OK. Nous sommes restés à 3 durant la finalisation du contrat de distribution (2-3 mois).

Sql Techn nous avais prêté un Ingénieur expérimenté en Gupta (!!) pour faire les premier avant-ventes. Comme cela ne fonctionnait pas trop, un vrai avant-ventes (de chez PC Tech), Daniel Gérard, nous a rapidement rejoint. Dès lors les avant-ventes étaient dignes de ce nom.
Nous arrivions sur le marché des L4G Groupware (re-Oupss !!) pas encore structuré ou se côtoyais des offres de langages (VB, C++…), des générateurs divers (Supernova, Infine…) et de vrais acteurs
de notre dimension en France qu'étaient SQLWindows (Gupta) et NSDK (NatSystem). Nous avions l'avantage d'être une société américaine avec des noms prestigieux comme références.

Nos concurrent directs disposaient de bases installée, certes auprès d'une population "Early adopter" mais dont les équipes "Veille techno" ou organisation (oui ca a existé) était quand même sous perfusion des éditeurs. Il a donc fallu déployer une stratégie à 2 vitesses.
1- Attaquer les comptes directs en faisant levier des références internationales de PowerSoft (Exxon, D&B, Bank of America, …)
2- Travailler les partenaires SSII pour quelles forment des compétences et nous positionne sur des projets en phase de déploiement auprès des grands ou moyens comptes.
Il a aussi fallu se faire connaitre auprès des cabinets d'études et de conseil qui était à l'époque plusieurs à rédiger des études comparatives des offres du moment. A cette occasion j'ai travaillé avec SQLI (à Marne la Vallée).

Le premier client significatif a été IP, l'agence pub du groupe RTL. Un contrat de 600k Francs ;-) 4 mois après mon arrivée. Le second BSA (Expertise en assurance) 350K Francs dans la foulée.
Ces 2 affaires ont été réalisées sur introduction de 2 SSCI qui ont à cette occasion récupérées les développements en régie. D'autres clients plus modestes sont rapidement venus rejoindre les premiers et les premiers appels entrants on commencé.
Plusieurs facteurs on concouru au succès que l'ont sait. Aux USA la boîte a vraiment cartonné. Version beta de PB 1.0 en aout 1990, version officielle en juillet 1991 (6 mois de vente, $5.2 million). PB V2.0 en Juillet 1992 ($22.1 million).
Le management fort expérimenté (Michael Kerzman & David Litwack) a su prendre les bonnes décisions au bon moment.
1- Accord entre M.KERZMAN et D. LITWACK 1 an après le départ de ce dernier du poste V.P. of R&D de Cullinet.
2- Changement de nom de la société Computer Solutions Inc (SSII et MRP) en Powersoft.
3- Revente du MRP GrowthPower (HP3000) à D&B pour financer le développement de PowebBuilder.
4- Programme ambitieux de beta tests (American Airlines, Microsoft, 3M, Fidelity Investments, Coca-Cola, ...).
5- Introduction au NASDAQ en février 1993.
6- Politique de soutien marketing et technique formidablement efficace.

7- Rachat d'éditeur de Base de données de compilateur (Watcom) et d'outil de conception. Innovation permanente et service client de grande qualité.
8- Fin 1993 Powersoft a exercée une option de rachat de ses distributeurs principaux, dont la France.
9- Novembre 1994, SYBASE annonce le rachat de Powersoft pour $870 Million pour un C.A de $133 million.

Plus généralement, quand les gros contrats on commencés à se bousculer (CNAV, EDF….) Powerbuilder est devenu le standard incontournable sur le marché français. En 1994, J'ai reçu la récompense (Award) de la meilleure performance mondial du groupe. Le lent déclin a commencé avec l'intégration pure et simple des équipes Powersoft dans le groupe Sybase (culture vraiment différente). Mais le coup de grâce a été le basculement des budgets des développements en 1996 au profit d'un nouvel acteur mondial, Internet.
=====

Phase 3, les RAD pour le Web
Cold Fusion est l'un des tous premiers outil de développement pensé et créé pour le Web par les frères Allaire en juillet 1995. Tout d'abord conçu pour simplifier la connection entre des pages HTML et des bases de données SQL. Par rapport aux outils client-serveur disponibles à la même époque, Cold Fusion version 1.0 (ainsi que la 1.5) était un produit simple, primitif même !
Cold Fusion évolua dès sa version 2.0 dans un environnement complet avec un langage de script ad hoc. Cold Fusion était développé avec Visual C++ ce qui le limitait de facto à tourner sous Windows NT (mais il eu tout de même un portage vers Sun Solaris à partir de la version 3.1 en janvier 98).
La petite société des frères Allaire fut rachetée par Macromedia qui elle même fut avalée par Adobe, ainsi va la vie des start-up : quand elles sont réussies, un plus gros se jette dessus, quand elles sont ratées, on les oublie vite !
Cold Fusion fut le premier mais pas le seul : un produit concurrent appelé DBWeb (développé par Aspect Software Engineering) tenta aussi sa chance en 1995 mais n'eu pas le même succès que le produit des frères Allaire. DBWeb évolua en i-Basic en 1996 qui fut alors remarqué par Microsoft qui le racheta en le transforma, ce qui donna MS ASP qu'on connait encore aujourd'hui…

Le succès de Cold Fusion résidait dans sa simplicité… Pour la démontrer, voici un exemple de code (pas d'affolement, c'est le seul exemple de code qu'on trouvera dans ce livre !) :

```
<!---********************--->
<!---First page in HTML--->
<form action = "resultspage.cfm" method = "post">
<center> <p> Enter your choice of search criteria for number of credits.
</p></center>
<center><input name = "user_choice_of_credits" type = "text" size =
"10"></center>
<br />
<center><input type="submit" value="Submit"></center>
<!---********************--->
<!---second page in ColdFusion--->
<center>
<cfquery name="ccc"  DATASOURCE="cbad725">
    select * from Course where Credits = #user_choice_of_credits#
</cfquery>
<cfoutput>
    <p>
```

```
        The #user_choice_of_credits# credit courses are:
        </p>
    </cfoutput>
    <cfoutput query="ccc">
        #Cno# - #Title# - #Credits# <br />
    </cfoutput>
    <cfoutput>
    <cfif ccc.recordCount EQ 0>
        There are no #user_choice_of_credits# credit classes.
    <cfelse>
        The number of #user_choice_of_credits# credit courses is
    #ccc.recordCount#.
    </cfif>
    </cfoutput>
    </center>
    <hr>
    <!---********************--->
```

Beaucoup plus ambitieux que Cold Fusion mais aussi beaucoup moins simple, NexT proposa WebObjects en mars 96. L'approche orienté objet de ce produit attira des clients comme Disney, Dell Computer et BBC News. Quand Apple absorba NexT en 97, WebObjects fut un peu délaissé mais c'est tout de même grâce à cet outil qu'Apple développa et fait tourner son "Apple Store" ainsi que la très connue "iTunes Store"...
Pendant que Cold Fusion s'affinait version après version, une autre bataille animait le marché : celle des serveurs d'applications basé sur Java. Bien qu'annoncé pour révolutionner le poste client, c'est surtout côté serveur que Java parvint à exister. Les serveurs d'applications basé sur ce langage étaient nombreux : Net Dynamics, Forté, WebLogic, Silverstream (sans compter les projets open source qui sont apparus par la suite comme Tomcat ou JBoss) et j'en oublie forcément !
Les frères Allaire racontent "Java est la meilleure chose qui pouvait nous arriver : pendant que tout le monde se jetait dessus, nous pouvions continuer à faire évoluer Cold Fusion, caché dans l'ombre de Java…". Et, effectivement, l'affrontement entre serveurs d'applications a monopolisé l'attention de la presse spécialisée pendant quelques années. Mais ce n'est pas pour autant qu'elle a débouché sur un produit stable, mature et bien répandu… Sun, par exemple, en a racheté plusieurs pour finalement ne pas en faire grand chose.

Les serveurs d'applications Java dispersés entre les grands acteurs, Cold Fusion racheté par Adobe, WebObjects par Apple, c'est finalement un tout petit projet qui mit tout le monde d'accord ces dernières années : PHP.

PHP ou le triomphe de l'Open Source
PHP signifiait "personal home page" quand il fut dévelopé en 1994 par Ramus Lerdorf, un programmeur Danois qui travaillait chez Greenlandic. Au début, ce n'était qu'une librairie écrite en C et qui s'appuyait sur la passerelle CGI. Lerdorf développa PHP pour remplacer un petit ensemble de scripts Perl qu'il utilisait alors pour gérer sa page Web personnelle. Cela lui permettait principalement de mesurer le trafic reçu par sa page. La première version s'appelait PHP/FI (FI pour Form Interpreter).
Lerdorf publia PHP en juin 95 afin d'accélérer sa mise au point. Zeev Suraski et Andi Gutmans, deux programmeurs Israéliens, réécrivirent le "parser" de PHP en 1997 et le renommèrent PHP: Hypertext Preprocessor. Cette nouvelle mouture de PHP (PHP/FI 2) fut publiée en novembre 97 après quelques mois de beta tests. La version 3 de PHP 3 suivit en juin 98 et nos deux compéres réécrivirent encore le noyau de PHP pour déboucher sur le "Zend Engine" en 1999. A partir de là, les deux programmeurs fondèrent la société Zend Technologies qui assure toujours la promotion et la diffusion de PHP.

La domination de PHP n'est pas tout à fait totale, on voit toujours apparaitre de nouveaux langages et de nouveaux environnements et certains deviennent même (pour un temps) à la mode. C'est par exemple le cas de "Ruby on Rails". Ruby on Rail (ou RoR) est un "framework" de développement Web issu de Basecamp, un très bon service en ligne dédié à la gestion de projet. La première version de RoR date de juillet 2004 (mais la vraie version 1 date de décembre 2005) et, depuis, sa popularité ne cesse de croitre (mais pas au point d'eclipser PHP, pas encore !).
En réaction et inspiré par les idées amenées par RoR, de nombreux framework censés faciliter le développement avec PHP on vu le jour. L'évolution des environnements de développement ne va pas s'arrêter là !

Quand on regarde cette évolution on retrouve une sorte de séries de "vagues".
Au début, les développeurs codent tout à la main, ce qui se traduit
vite par un goulet d'étranglement. D'où le besoin d'un langage de plus haut niveau (l'exemple du problème de l'ingénieur évoqué avec NOMAD est un exemple significatif). Mais après une vague d'euphorie, on redescend un peu sur Terre. Car un langage de trop haut niveau ne permet pas autant de customisation (et aussi, se révéle très gourmand en ressources machine, d'où des performances souvent décevantes en exécution). On se rabat donc sur des langages "intermédiaires". Un langage qui nécessite plusieurs lignes de code pour résoudre le problème de l'ingénieur.

PHP est l'exemple type de cette "oscillation". Il est de plus bas niveau que Cold Fusion, mais le problème avec le code mis en illustration est qu'il représente un cas d'exemple qui n'arrive que trop rarement dans la réalité : afficher tel quel le contenu d'une requête SQL. Dans le contexte réel, une logique métier complète est souvent appliquée à la présentation… Par exemple, on va traiter chaque colonne différemment. Si la valeur est en-dessous d'un certain seuil on affiche en rouge. On affiche la valeur sauf si le champs X ou le champs Y ne sont pas nuls, et ainsi de suite… Mais au bout du compte on facilite quand même la tâche du programmeur.
Mais étant donné que l'informatique tend à se complexifier, les développeurs redeviennent le goulet d'étranglement, d'où la recherche de nouvelles méthodes de plus haut niveau (les ERP et les CMS deviennent les nouveaux environnements de développement de référence pour les applications d'entreprises).

Les ERP justement, parlons-en avec l'histoire de SAP, l'acteur symbole de ce secteur...
Trois éditeurs ont symbolisé le tournant des années 90 : Computer Associates, Oracle et SAP. Ces trois-là ont incarné les principaux phénomènes qui ont façonné le marché des logiciels (en dehors de la vague des logiciels individuels sur PC) : la consolidation financière (en clair, CA rachetant toutes les proies faciles possibles), l'avénement des bases de données relationnelles (avec Oracle dominant le marché devant Sybase et Informix) et la montée des ERP (*Entreprise Ressources Planning* ou PGI pour progiciels de gestion intégrés).

Le développement commercial des progiciels posait le problème de la diversité des systèmes et des bases de données au sein du systèmes d'informations des entreprises clientes. Les éditeurs de logiciels spécialisés par secteur proposaient de résoudre ce problème évident par une solution évidente : prenez tout chez moi !
C'est ainsi que l'on vit quelques éditeurs devenir dominants dans leur secteur de référence. Mais cette compartimentalisation du marché ne pouvait durer et le premier à venir avec une solution universelle (ou, plus exactement, se définissant comme universelle) allait remporter la mise. Contre toute attente, c'est l'allemand SAP (et dans une moindre mesure le néerlandais Baan) qui allait réussir cette perçée. L'ERP était la solution que le marché attendait : ce n'était pas vraiment un progiciel fin prêt à être utilisé mais plutôt un cadre de développement à paramétrer selon les besoins et l'organisation du client.
C'est la très connue "crise du bug de l'an 2000" qui acheva de généraliser les ERP dans toutes les grandes organisations : plutôt que de tenter de corriger les grandes chaines de traitements qui, souvent, dataient des années 70 (voire des années 60 pour certains

et dans bien des cas, le code source avait disparu depuis un bon moment !), le plus simple était de basculer sur un ERP pour retrouver une situation "claire"...

Voici comment une minuscule SSII allemande est devenue en quelques décénies le premier éditeur de solutions d'entreprises au monde :

Le parcours accidentel mais fructueux de SAP
SAP fut fondée en avril 1972 par cinq anciens employés d'IBM Allemagne. Les cinq associés quittèrent IBM car ils étaient motivés par la proposition d'ICI (Imperial Chemical Industries, le groupe chimique britanique bien connu était alors client d'un mainframe IBM 370 et voulait que soit développé un logiciel de gestion de production et de comptabilité) qui venait d'être rejetée par IBM Allemagne qui ne voulait pas sortir de son rôle de constructeur. ICI proposa le contrat aux cinq "conjurés" qui virent ainsi l'opportunité de créer leur société avec un projet qui pourrait être proposé à d'autres clients par la suite.
Ce que ICI voulait était un embryon de logiciel de gestion de production inspiré de la notion de MRP très en vogue dans les années 70 (comme on l'a vu avec ManMan d'ASK).

==== encadré sur MRP/ERP ========
Voilà ce que nous dit wikipedia sur les notions de MRP et d'ERP

On distingue le MRP (Material Requirements Planning), né formellement aux Etats-Unis en 1965 et qui ne représente alors qu'une méthode de calcul des besoins en matières, de son évolution, le MRP2 ou MRP II (Manufacturing Resources Planning). Ce modèle plus large, qui intègre la gestion de toutes les ressources de l'entreprise (consommables, c'est-à-dire « matières et composants », et renouvelables, c'est-à-dire « capacité machines et main-d'œuvre »), constitue un système de pilotage des ressources qui repose sur la prévision des ventes et les nomenclatures de produits et qui opère comme le MRP en flux poussé (c'est-à-dire que l'on établit le plan de production sur la base de prévisions).
Le MRP transforme en données de production les données commerciales relatives aux ventes. Le MRP2 intègre d'autres fonctions telles que la planification à capacité infinie, l'ordonnancement à capacité finie des ressources machine et main d'œuvre, le suivi de production, le calcul des coûts, etc.
Le MRP2 est le précurseur des progiciels ERP (Enterprise Resources Planning) et reste souvent l'un de leurs modules fondamentaux.

Le terme ERP provient du nom de la méthode MRP (Manufacturing Resource Planning) utilisée depuis les années 1970 pour la gestion et la planification de la production industrielle.
Le principe fondateur d'un ERP est de construire des applications informatiques (paie, comptabilité, gestion de stocks…) de manière modulaire (modules indépendants entre eux) tout en partageant une base de données unique et commune. Cela crée une différence importante avec la situation préexistante (les applications sur mesure existant avant les ERP) car les données sont désormais supposées standardisées et partagées, ce qui élimine les saisies multiples et évite (en théorie) l'ambiguïté des données multiples de même nature (exemple : société TRUC, TRUC SA et Sté TRUC…).
L'autre principe qui caractérise un ERP est l'usage systématique de ce qu'on appelle un moteur de workflow (qui n'est pas toujours visible de l'utilisateur), et qui permet, lorsqu'une donnée est entrée dans le système d'information, de la propager dans tous les modules du système qui en ont besoin, selon une programmation prédéfinie.
Ainsi, on peut parler d'ERP lorsqu'on est en présence d'un système d'information composé de plusieurs applications partageant une seule et même

base de données, par le biais d'un système automatisé prédéfini éventuellement paramétrable (un moteur de workflow).
==========

La coopération avec ICI a été fondamentale dans les débuts de SAP : non-seulement ICI permettait aux cinq fondateurs de développer le logiciel sur le mainframe de la société mais le contrat prévoyait que le produit final pourrait être proposé à d'autres clients sans qu'ICI ne touche de royalties. ICI alla même jusqu'à arranger des visites clients afin de faciliter la prospection de SAP (comme anciens d'IBM Allemagne, nos cinq fondateurs avaient accès aux vaste réseau de clients d'IBM, leur cible de base pour la commercialisation de leur logiciel).
La première version du logiciel alors appelé MIAS (pour Material Information and Accounting System) fut terminée en janvier 1973 mais des modules supplémentaires furent développées l'année suivante. La commercialisation progressait également de façon satisfaisante : en 1975, MIAS était installé sur 40 sites et ce nombre atteignait 100 en 1978.
En 1977, SAP porta son logiciel sur mainframe Siemens avec lequel il y eu un accord de commercialisation mais le focus sur les systèmes IBM restait primordial.
En 1979, SAP put enfin acquérir son propre mainframe IBM et en profita pour se lancer dans le développement d'une nouvelle version de son logiciel enfin appelé R/2 qui put être lancé en 1981. Pour se conformer à la norme MRP-II, R/2 contenait des modules supplémentaires dont certains comme le module de RH avait été développé en coopération avec des clients. La phase suivante du développement a été l'internationalisation du logiciel et sa commercialisation.
Là encore, il ne s'agit pas de l'exécution d'un plan soigneusement défini à l'avance mais plutôt d'une opportunité apportée par les clients et que SAP n'a fait que suivre (mais le mérite n'est pas nul : les dirigeants auraient pu rester aveugles et sourds à ces opportunités et les rejeter les unes après les autres… comme d'autres l'ont fait !). Cette fois, c'est la direction européenne de John Deere qui décida d'installer R/2 dans toutes ses usines en Europe et cela nécessitait sa traduction en anglais et en français. SAP créa une filiale en Suisse en 1984 pour accompagner ce développement à l'international.
Les concurrents américains de SAP tentèrent également de s'implanter en Europe mais il s'avéra que le chemin était plus difficile pour eux que pour un européen qui était déjà habitué aux nombreuses contraintes locales (monnaies, législations et autres). De plus, les éditeurs américains bénéficiaient du confort d'un marché domestique vaste, ce qui reléguait la conquête du "reste du monde" à toujours plus tard…
Au fur et à mesure de son expansion, SAP réussit à déclencher un autre relais de croissance sans tomber dans le piège classique de l'intégration verticale : le besoin de personnel compétent pour accompagner les installations et paramétrage de R/2 allant croissant, au lieu d'essayer d'embaucher des consultants en interne, SAP sut s'appuyer sur des sociétés extérieures en tissant un réseau de partenariats. Cette démarche fut particulièrement fructueuse quand ce réseau s'étendit au-delà des SSII classiques pour toucher aussi ce qu'on appelait alors les "big six" : Price Waterhouse, Coopers & Lybrand, Deloitte & Touche, Arthur Andersen, KPMG et Ernst & Young, les sociétés de consultants spécialisées dans l'audit et la comptabilité. (ces "big six" sont aujourd'hui devenus les "big four" suite à des fusions et acquisitions entre eux…). Ces consultants avaient un accès fréquent aux directions générales (bien plus que les SSII) et servirent de relais commercial très efficace pour diffuser la bonne parole en faveur de R/2… Car SAP a toujours fait très attention à partager les recettes avec ses partenaires, au lieu de vouloir s'accaparer tous les bénéfices.

Le tournant vers R/3
Dans les années 1987/88, SAP s'introduisit à la bourse de Francfort mais c'est aussi lors de cette période qu'IBM, son principal partenaire technique lança son offre AS/400 ainsi que ses premiers systèmes Unix (la première station de travail Unix de big blue, la 6150 avait été dévoilée en 1986). Influencé par IBM et par un client (les chemins de fer

allemands, la Deutsche Bahn), SAP se lança dans le développement d'une version de son logiciel pour l'AS/400 : R/3. Initialement, R/3 n'était pas du tout destiné à remplacer R/2 mais plutôt pour attaquer le marché des PME. Les sociétés moyennes n'étaient évidemment pas équipées avec des mainframes et un mini comme l'AS/400 leur était directement destiné.
Le développement de R/3 a été plus que difficile puisqu'il prit quatre ans (de 1987 à 1991) et failli être annulé à deux mois de son annonce au CEBIT de 1991… Il s'avéra que R/3 était trop exigeant en ressources pour tourner sans mal sur un AS/400 !
Et la solution fut plutôt de le proposer sur serveur Unix. En fait, le développement avait eu lieu sur Unix et avec un SGBDR Oracle, R/3 fonctionnait convenablement. Ce tournant représentait beaucoup pour une société qui avait jusque-là vécu quasiment exclusivement dans l'ombre d'IBM mais il s'avéra être la bonne décision. En Europe, les ventes de R/3 démarrèrent doucement mais la perçée eu lieu aux USA : présenté à l'occasion de la réunion Sapphire (le groupe utilisateurs des clients américains), R/3 emporta l'enthousiasme des grandes sociétés américaines qui étaient déjà au fait du modèle client-serveur utilisé par R/3.
Le chiffre d'affaires de SAP explosa principalement grâce à la croissance de ses ventes aux USA. Accessoirement, cela aida aussi Oracle à consolider sa position de SGBDR de référence. La collaboration avec Oracle est progressivement devenue une cohabitation au fur et à mesure que le fournisseur de SGBDR devenait un concurrent direct (au fil des années, Oracle a racheté PeopleSoft et J.D. Edwards, deux ex-concurrents de SAP). Du coup, SAP chercha à se défaire de cette "dépendance" vis-à-vis d'Oracle.
Courant 1998, les spéculations et rumeurs allaient bon train sur le SGBDR que SAP pouvait ou allait racheter. Sybase et Informix (ce dernier va être racheté par la suite par IBM) étaient les plus souvent cités car, même si les installations de l'ERP SAP/R3 reposaient majoritairement sur Oracle comme base de données, l'opposition et la concurrence entre les deux éditeurs était vive. Il apparaissait donc naturel que SAP veuille pratiquer l'intégration verticale en rachetant un éditeur de SGBDR afin de proposer sa base de données plutôt que celle d'Oracle par ailleurs éditeur d'ERP lui aussi. Finalement, SAP s'est contenté de passer un accord avec Software AG autour du SGBD Adabas. A travers cet accord, SAP obtenait le droit de développer et de revendre sa propre version d'Adabas renommée SAP DB pour l'occasion. Mais cet accord eu peu d'effets par la suite et, encore aujourd'hui, au moins les deux tiers des installations de SAP R/3 tournent avec Oracle comme SGBDR.
SAP ne s'est pas retrouvé confronté qu'à Oracle comme ancien partenaire technique… Microsoft qui affirmait (par le biais d'une déclaration de Bill Gates) qu'il ne "ferait jamais de logiciel de comptabilité" dans les années 90 a évolué depuis : fin 2000, Microsoft racheta Great Plains Software qui développait et commercialisait un logiciel de compta justement… En 2002, récidiva avec le rachat de Navision qui proposait un ERP. Ces acquisitions furent rassemblées par la suite dans la division Microsoft Business Solution.
Le succès de SAP, lui, a grandement été du au fait que la compagnie n'a pas été trop gourmande. Contrairement à Microsoft qui propose un partenariat intéressant au court terme mais tente ensuite d'accaparer tous les profits au long terme (contrat OEM léonins proposés aux fabricants de PC par exemple), SAP s'est toujours assuré que ses partenaires aient une bonne part du gâteau (de l'ordre de 60% de ce que dépensent les entreprises pour et autour de SAP).
Au final, tout le monde s'y retrouve: SAP, ses partenaires qui font des gros profits en conseil, le directeur informatique (un gros projet justifie de grosses ressources -on estime que SAP R/3 coûte 5 à 10 fois plus cher à installer qu'à acheter-, et SAP est apparemment très stable). Seul l'utilisateur final ne s'y retrouve pas (l'interface utilisateur est horrible, mais l'utilisateur final a rarement la parole).

Mais SAP reste le leader de ce secteur et continue son expansion dans tous les sens avec le rachat spectaculaire de Business Objects (l'éditeur français d'outils d'infocentre) en 2008 (opération annonçée en octobre 2007).

Il est normal de parler SAP pour évoquer la montée des ERP mais il serait injuste d'oublier un autre acteur important de cette épopée... Baan.

=========

L'histoire de Baan racontée par Joseph Gonzalez - Business Solutions Leader à IBM Global Business Services

Baan a été créée en 1978 aux Pays Bas par Jan Baan, comptable de profession, qui avait perçu le besoin d'utiliser des logiciels dans le domaine de la gestion de production. Au départ, la société est spécialisée dans le consulting.
La plupart des salariés de Baan étaient membres de l'Eglise Réformée Néerlandaise. Les jupes (longues) étaient de rigueur pour les femmes et travailler le dimanche était interdit.

A la fin des années 80, les entreprises commencent à repenser et optimiser l'ensemble de leurs processus. Jan Baan décide de développer une application de gestion assemblant des logiciels spécialisés afin de traiter toutes les données de l'entreprise, de la production au commercial et à la comptabilité. Le concept de l'ERP était né. Le progiciel s'appelle TRITON. Paul Baan, le jeune frère de Paul s'occupe de Baan en Hollande.

Après un démarrage rapide, la crise du marché en 1987-1990 porte un coup rude à la société qui peine à survivre.
En 1990, Jan Baan confie à Kees Westerhuis le développement commercial de Baan afin de doter la société des alliances nécessaires à une expansion internationale et à l'attraction des capitaux nécessaires au développement.

En Juin 1991, un contrat OEM est signé avec la société Bull, au niveau mondial, qui donne à Bull la possibilité de construire sa propre offre ERP sur la base de TRITON, en même temps que ce contrat apporte à Baan une manne d'argent frais qui la sauve de faillite. En parallèle, Baan signe un autre contrat OEM avec ASK Software ainsi que des contrats de revente avec IBM et avec la société Focal en France. D'autres contrats de revente sont signés dans d'autres pays en Europe. Evitant toute exclusivité, Kees Westerhuis dote Baan d'un réseau de revente dans les principaux pays d'Europe. En parallèle, Laurens van der Tang, dirige la R&D, et établit des liens avec l'Inde pour y localiser les développements.

Bull dénomme son progiciel ERP «STRATEGE» et en fait son fer de lance pour gagner de nouveaux clients. Bull me confie le rôle de développer le business STRATEGE en France en Juillet 1991.
Ma première réunion avec Baan dans les bureaux de Bull à La Défense (Tour Bull) est un évènement douloureux : mon anglais scolaire, que je n'ai pas utilisé depuis 20 ans ne tient pas la route au cours d'une discussion business aux enjeux importants.
Dans le mois qui suit je suis en formation pour 2 semaines au siège de Baan, à Ede (NL). Les bureaux modernes sont nichés dans la campagne néerlandaise, dans un environnement boisé. La salle de formation est à l'étage du bâtiment et pour y accéder je passe devant le bureau de Jan Baan que je croise alors pour la première fois.
La salle de formation est pleine de partenaires de tous les pays : Anglais, Espagnols, Italiens, Français (moi), Allemands, Hollandais. Et Ecossais : deux écossais qui arrivent en retard et s'assoient à la table devant la mienne et explique, avec leur accent rocailleux inimitable qu'ils sont en retard à cause d'un «TRRRaffic Jam».
C'est lors de cette formation que je fais la connaissance de Jan Baan, Laurens van der Tang, Otto van der Tang (Channel Director) et Andrei Bottema (VP en charge de la Vision et de la Stratégie).

C’est aussi cette formation qui sera le déclencheur d’une pratique courante de l’anglais : à la fin de la première semaine de formation, lorsque je commence à rêver et faire des cauchemars en anglais, je me dis que c’est bon, je parle couramment anglais maintenant.

De retour en France, j’ai à gérer un aspect imprévu : Baan avait une filiale en France, Baan France, qui a fermé ses portes mi 1991, juste avant le contrat OEM avec Bull. Nous recevons la candidature de plusieurs ex-salariés. Nous en recrutons 2 chez Bull : Pascal Echardour et Marc Billant. Pascal devient commercial dans mon équipe et Marc chef de projet dans l’équipe de mise en oeuvre.

Très vite, les ventes de STRATEGE démarrent : d’abord des PME (ESG, Roset, Comatelec,...), puis de sociétés de plus en plus importantes (Schlumberger, GIAT, ...). Alors que SAP est cantonné chez les très grands comptes avec son R2 rigide et finance focus, le marché est alors enthousiasmé par l’offre de Baan. en France, la quasi totalité des ventes sont faites par mon équipe.
En 1993, de plus en plus, les clients choisissent STRATEGE de Bull, mais exigent des plate-formes matérielles non Bull. Je suis salarié de Bull. Je dois choisir : satisfaire l’attente des clients ou me contenter de vendre du 100% Bull.
Je choisis de quitter Bull fin 1993 et de créer ma propre société pour ... devenir revendeur de Baan en France. Prétentieux !
Baan est en discussion avec plusieurs acteurs français (dont Bull) pour créer sa filiale Française.
En décembre 93, je présente mon projet au directeur dont dépend mon entité, André Marreck : il me dit poliment «au revoir».
Je présente mon projet à Jacques Wauquier (Associé chez KPMG Peat Marwick France) qui est séduit et organise un déjeuner avec son PDG, également séduit : 2 heures après le déjeuner il me remet le chèque des 47% du capital que je lui ai proposé.
Ainsi, je crée «Genesys Manufacturing» et m’attelle dès Janvier 1994 à bloquer les négociations de Baan avec ses partenaires actuels pour créer Baan France. Six mois de négociations avec Otto Van Der Tang et Kees Westerhuis. Six mois plus tard, Genesys Manufacturing signe un contrat de revendeur avec Baan. Les ventes explosent (Gec Alsthom, des PMEs, ...).
General Atlantic entre au capital de Baan en 1993 pour préparer une entrée en Bourse. En 1994 Baan signé avec Boeing un contrat de 25 Millions USD.
A cette époque, SAP est quasi inexistant sur le marché. On croise le fer avec MFGPRO, Movex. Au premier trimestre 1995, Baan qui prépare son introduction au Nasdaq, nous propose de prendre la totalité du capital de Genesys Manufacturing. Lorsque je rencontre Jan Baan à Ede pour négocier les conditions du rachat, une belle maquette d’un Boeing 777 trône sur son bureau.
Jackpot : la cession à Baan est signé en Mars 1995. Quelques jours avant l’introduction au Nasdaq. Genesys Manufacturing devient Baan France et j’en deviens le PDG, avec un gros paquet de stock options en plus de la cession de mes 53% de Genesys Manufacturing.
La création de Baan France est officialisée en Mars 1995 lors d’une conférence de presse de Jan Baan au cours du salon ERP de Mars 1995 à Villepinte.
Road Show mondial de Baan pour son introduction au Nasdaq : à Paris, cela se passe au Crillon. Jan Baan et David Cairns (CFO) se sont déplacés. Jan Baan traverse la salle remplie de banquiers et investisseurs, se dirige vers moi et me serre dans ses bras. David Cairns me salut d’un «hi, do you meet your budget». Puis les deux s’en vont saluer les financiers. Au déjeuner, je suis impressionné : je n’ai jamais vu autant de banquiers partager un repas avec moi !
TRITON devient BAAN. en majuscule, pour différencier avec la société Baan Company.

Sky Rocket : pour décrire le business de Baan France de 1995 à fin 1996, pas de meilleur terme.
Je me souviens du salon ERP de l'automne 1995 : SAP prend un énorme stand sur ce salon. Baan France c'est seulement 20 personnes à cette époque. Agiles, nous choisissons les 4 stands aux angles du salon et le stand du centre. Les visiteurs ne peuvent faire un pas sans tomber sur un stand Baan. Baan est partout. SAP est dépité du peu de visiteurs sur son stand. C'était l'époque où les salons avaient encore un rôle important dans le business.
Alstom, ABB, Schneider, Otis, Snecma, Carrier, Dassault Electronics, Philips CS, Sensormatic, ... : au moins 1 «new name» par semaine, PME ou Grand Groupe.
La force de BAAN : un vrai ERP, avec une grande force dans le manufacturing, et une excellence dans la gestion à la commande (machines spéciales, etc...).
L'autre force de BAAN : la solution est verticalisée : Machines Spéciales, Electronics, Process, etc... et donc un impact fort chez les prospects.

60 salariés chez Baan France et 200 consultants dédiés chez les partenaires ; Bull (devenu notre revendeur lorsque Genesys Manufacturing devient Baan France), Origin, KPMG, Cap Gemini, E&Y, IBM, ...).
Ernst Jilderda est le VP Europe de Baan Company.

SAP devient un peu plus présent fin 1996 ; nous l'avons enfin eu en concurrence sur Schneider et Snecma.... La vie devenait ennuyeuse sans eux... ;)

1995-1996 : années des acquisitions pour Baan. Se succèdent Meta4 (HR), Aurum (CRM), Antalys (configuration software), Berclain (Moopi Synchronization & Scheduling software), CODA (Finance), etc...
En 1996, lancement de D.E.M. Dynamic Enterprise Modeling, outil de modélisation des processus qui génère automatiquement le paramétrage du progiciel BAAN
Encore aujourd'hui cette fonction est unique !

En 1996, Tom Timsley prend la Direction de Baan Company. Tom est un ancien de McKinsey chez qui il a oeuvré pendant 18 ans. Rien à voir avec le charisme et l'attitude visionnaire et entrepreneuriale de Jan Baan. Tom est un consultant de haut vol. Un monde nouveau démarre. Les yeux rivés sur le cours de l'action.

Je préfère quitter Baan Company : en meeting à Barneveld au nouveau siège de Baan, je l'annonce à Jan Baan qui m'assure de son soutien pour mes nouvelles entreprises. Puis je souhaite l'annoncer à Tom Timsley : il me répond «good bye».

Avant mon départ je négocie un contrat de revendeur Baan pour ma nouvelle société.

En Janvier 1997, je crée Genesys Enterprise Solutions qui en 18 mois devient l'un des 5 principaux revendeurs-intégrateurs de BAAN en France.
En Mars 1998, je cède cette société à TEAMLOG, et elle est renommée Teamlog Enterprise Solutions (T.E.S) dont je conserve la direction. Quelques mois plus tard, je participe à l'introduction de Teamlog au Second Marché de la Bourse de Paris en 1998 : lors du Road Show, je suis surpris et flatté que la majorité des questions des investisseurs concernent le business de T.E.S.

En Août 1998, l'action de Baan Company chute en Bourse. Rachat par Invensys. Rechute. Plus tard, rachat par SSA. Tout cela était prévisible depuis fin 1996.

Quand la seule référence c'est le cours de l'action, on ne pense plus au client, on perd contact avec la réalité, et toute la société est axée sur le court terme.

J'ai eu beaucoup de plaisir de travailler avec Robin Morin et Georges Beaume les dirigeants de Teamlog à cette époque.
En Juillet 1999, je suis, par le biais de ma holding, le plus gros actionnaire du Groupe Teamlog en dehors des fondateurs et des banques et fonds d'investissement.

Je décide de tourner la page Baan, je vends toutes mes parts et m'exile aux USA comme co-fondateur et EVP d'APOGON Corporation (un éditeur de logiciel SaaS en avance de 10 ans sur son temps). Nos bureaux sont à Reston, Freedom Drive, et donne sur la place centrale de Reston, avec vue sur le bureau de .... Tom Timsley, CEO de Baan Company, dont le headquarter est sis à la même adresse .....

Depuis j'ai revu et retravaillé avec Jan Baan, Paul Baan, Kees Westerhuis, Otto van der Tang, Ernst Jilderda, et j'anime toujours le Club des Anciens de Baan en France. Clin d'oeil : il y a pas mal d'anciens de Baan chez SAP...

=========

La vague Unix
Nous venons de l'évoquer de nombreuse fois ci-avant : Unix a été instrumental dans la montée et la diffusion des SGBDR et de SAP. Unix a proliféré dans les universités américaines principalement parce qu'il était diffusé gratuitement par AT&T. AT&T ne se lançait pas dans le marché informatique à cause des régles anti-trust américaines. La relative généralisation d'Unix sur quasiment toutes les plates-formes facilita grandement la communication entre les machines.
En 1985, on a évalué qu'Unix faisait tourner plus de puissance informatique que n'importe quel autre système… Unix a représenté un tournant sur bien des points mais c'est surtout le premier système d'exploitation qui ne venait pas d'un constructeur et qui s'est bien diffusé ET qui avait été conçu dès le départ pour être portable. Dans l'évolution du logiciel, Unix représente plus qu'une étape mais bien un des piliers de l'époque moderne...

Voici une brève histoire d'Unix, extraits issus de http://www.tuteurs.ens.fr/unix/histoire.html, de http://fr.wikipedia.org/wiki/UNIX et de http://www.linux-france.org/article/these/hackers_history/fr-a_brief_history_of_hackerdom.html

Les origines d'Unix remontent à la fin des années 1960 et à un laboratoire d'AT&T (à l'époque, la compagnie nationale de télécommunications américaine) appelé « Bell Labs ». Ce labo, en collaboration avec le MIT, développait un OS appelé « Multics ».
Multics fut un banc de tests pour des idées importantes, comme la manière dont on pouvait dissimuler la complexité d'un système d'exploitation au coeur de ce dernier, sans rien en laisser transparaître à l'utilisateur ni même à la plupart des programmeurs.
Les laboratoires Bell se sont retirés du projet quand Multics a montré des signes de boursouflement superflu (ce système a plus tard été mis sur le marché par la société Honeywell mais n'a jamais connu de succès).
Ken Thompson regrettait l'environnement de Multics, et a commencé à implanter en s'amusant un mélange des idées de Multics et de certaines des siennes propres sur un DEC PDP-7 qu'il avait sauvé du rebut.
Il écrit alors un jeu, « Space Travel » (« voyage spatial »), d'abord sur Multics, puis sous d'autres systèmes d'exploitation. Sur sa lancée, il écrit, en collaboration avec Dennis Ritchie, un système de gestion de fichiers, quelques utilitaires basiques comme cp ou rm et un interpréteur de commandes appelé shell.

Le nom Unics fut suggéré par Brian Kernighan en 1970 suite à un jeu de mot « latin » avec Multics; « Multi- car Multics faisait la même chose de plusieurs façons alors qu'Unics faisait chaque chose d'une seule façon ». Ce nom fut par la suite contracté en Unix (pour au final être déposé sous le nom UNIX par AT&T), cependant personne ne se souvient de qui est à l'origine de la modification du « cs » en « x ».
L'essor d'Unix est très fortement lié à un langage de programmation, le C. À l'origine, le premier Unix était écrit en assembleur, puis Ken Thompson crée un nouveau langage, le B. En 1971, Dennis Ritchie écrit à son tour un nouveau langage, fondé sur le B, le C. À l'instar d'Unix, C était conçu pour être agréable, sans contraintes, et souple. Aux laboratoires Bell, le mot a circulé, et ces outils ont attiré l'attention, jusqu'à être renforcés, en 1971, par une prime remportée par MM. Thompson et Ritchie, pour produire ce qu'on appellerait maintenant un système d'automatisation de bureau pour usage interne.

Dès 1973, presque tout Unix est réécrit en C. Ceci fait probablement d'Unix le premier système au monde écrit dans un langage portable, c'est-à-dire autre chose que de l'assembleur (l'assembleur est un langage très proche de la machine, compris directement par le processeur, il est donc particulier à chaque type de machine). Traditionnellement, les systèmes d'exploitation avaient été écrits en langage d'assemblage, ardu, pour fonctionner le plus rapidement possible sur leurs machines hôtes. MM. Thompson et Ritchie furent parmi les premiers à comprendre que le matériel et les techniques de compilation avaient fait suffisamment de progrès pour permettre d'écrire tout un système d'exploitation en langage C, et en 1974 tout l'environnement avait été porté avec succès sur plusieurs machines de types différents.
Cela n'avait jamais été réalisé auparavant, et les implications étaient énormes. Si Unix pouvait présenter le même visage, les mêmes possibilités, sur des machines de nombreux types différents, il pourrait servir d'environnement logiciel commun à toutes ces machines. Les utilisateurs ne souffriraient plus des coûts des nouvelles conceptions de logiciels chaque fois qu'une machine deviendrait obsolète. Les hackers pourraient transporter des boîtes à outils logicielles d'une machine à l'autre, plutôt que de devoir réinventer la roue et l'eau chaude à chaque fois.
En plus de leur caractère portable, Unix et C avaient d'autres atouts dans leur manche, et pas des moindres. Tous deux avaient été construits en suivant la philosophie du « Keep It Simple, Stupid » (acronyme signifiant « embrasser » et dont la version développée conseille de faire les choses simplement, sans prétentions). Un programmeur pouvait facilement retenir la totalité de la structure logique du C (à la différence de la plupart des autres langages, antérieurs ou postérieurs) sans devoir se référer sans cesse à des manuels ; et Unix était structuré comme une boîte à outils souple de programmes simples mis au point dans le but de se combiner utilement les uns avec les autres.
L'essor
Un décret datant de 1956 interdisait à l'entreprise AT&T, dont dépendait Bell Labs, de commercialiser autre chose que des équipements téléphoniques ou télégraphiques. C'est la raison pour laquelle la décision fut prise en 1973 de distribuer le système UNIX complet avec son code source dans les universités à des fins éducatives, moyennant l'acquisition d'une licence au prix très faible.
Chacun étant libre de développer des nouveautés, très vite apparaissent des familles différentes d'Unix. On peut regroupes les premiers clients d'Unix en deux groupes, les universités et centres de recherches américains d'une part, les grands constructeurs informatiques d'autre part.
Pour les universités et centres de recherche, Unix était un système peu onéreux (AT&T le leur vendait à un prix symbolique) et puissant. Encore maintenant, beaucoup utilisent Unix.
L'Université de Californie à Berkeley (UCB) notamment est à l'origine de l'une des plus anciennes branches d'Unix, BSD (Berkeley Software Distribution). C'est au début de 1977, Bill Joy réalise la première Berkeley Software Distribution.
Plus tard, avec l'arrivée de nouveaux terminaux, il écrit vi (l'éditeur visuel), une surcouche de ex. La Second Berkeley Software Distribution ou 2BSD voit le jour à l'été 78. Puis en décembre 1979, Bill Joy distribue la 3BSD, la première qui supporte les

ordinateurs VAX de DEC. Unix atteint alors sa version 7, son évolution s'accompagnant de nombreuses modifications notables telles que l'extension à 2 Go de la taille maximale d'un fichier, l'ajout de plusieurs utilitaires, et surtout la portabilité du système. C'est à cette époque que le premier grand portage d'UNIX, la version 32/V, fut réalisé, sur un VAX 11/780.
Lors de la publication de 3BSD en 1979, la Defense Advanced Research Projects Agency (DARPA) prend connaissance des avancées réalisées à l'UCB. Ils ont l'intention d'utiliser UNIX pour leurs projets. En automne de cette même année, Bob Fabry propose à la DARPA une version augmentée de 3BSD pour répondre à leurs besoins. Un contrat de 18 mois est signé en avril 1980, et Bob Fabry rassemble une équipe. Bill Joy, qui vient juste de passer sa soutenance de thèse (doctorat), se propose de participer. Les versions se succèdent jusqu'à 4.1BSD. Satisfaite, la DARPA signe pour deux ans supplémentaires et le budget est presque multiplié par cinq.

L'autre grande branche d'Unix est Unix System V (lire « cinq »), vendu par AT&T aux grands constructeurs de matériel comme Sun Microsystems ou Hewlett-Packard. Le tournant est venu au milieu des années 80 quand même IBM s'est décidé à proposer sa propre version d'Unix avec Aix. AIX avait la réputation parmi ses utilisateurs d'être incohérent avec les autres systèmes Unix. Une plaisanterie habituelle était que son acronyme signifiait « Ain't unIX » (ce n'est pas Unix). Dans la pratique, UNIX avait déjà tellement de variantes panachées de BSD et de System V que peu de personnes savaient vraiment où était le standard, ni même seulement si quiconque en avait défini un. Il faut dire que les versions d'Unix étaient nombreuse… Dès 1977, AT&T mit les sources d'UNIX à la disposition des autres entreprises, si bien qu'un grand nombre de dérivés d'UNIX furent développés :
- XENIX, fondé sur la 7e édition développé en 1980 par Microsoft.
- AIX, développé par IBM, dont la première version de 1986 fut basée sur System V release 2.
- Solaris (d'abord appelé Sun OS), développé par Sun Microsystems, basé au départ sur BSD 4.1c en 1981, puis sur System V release 4 (SVR4).
- HP-UX, fondé sur System V, développé à partir de 1986 par Hewlett-Packard
- Ultrix, développé par DEC. La version Ultrix-11, destinée aux machines de la famille PDP-11, est basée sur la 7e édition, avec des ajouts provenant de System V et de BSD. La version Ultrix-32, destinée aux machines de la famille VAX, est essentiellement fondée sur BSD.
- IRIX, développé par SGI depuis 1986.
- UnixWare, descendant de SVR4, développé par Novell puis revendu à SCO Group.
- SCO Group UNIX, fondé sur XENIX et System V développé dès 1979 par Santa Cruz Operations et Hewlett-Packard.
- Tru64, fondé sur une version du micro-noyau Mach 2.5 réalisée par le consortium OSF (Open Software Foundation). Il a d'abord été développé sous le nom OSF/1 puis DEC UNIX par Digital Equipment Corporation, Compaq et enfin Hewlett Packard.
- A/UX, un UNIX développé par Apple, compatible avec Mac OS.

Chacun de ces grands groupes a développé Unix selon ses propres besoins et intérêts, créant ainsi son propre Unix. Cette prolifération entraina forcément une grande confusion et les acteurs du marché sentaient bien que pour qu'Unix devienne vraiment un système d'exploitation reconnu et utilisé partout, il fallait le standardiser…
Cette situation conduisit des membres du groupe d'utilisateurs /usr/group, qui a pris depuis le nom de UniForum, à forger un standard UNIX dès 1981 afin d'assurer une portabilité maximale entre les différents systèmes : en 1984 le groupe /usr/group publie POSIX, une série de standards développés sous couvert de l'IEEE (Institute of Electrical and Electronics Engineers). POSIX est ainsi également connu sous le nom IEEE P1003.
En 1985, AT&T publie SVID (System V Interface Definition) décrivant le System V. Cette première définition est différente de POSIX.
À la même époque, un consortium de constructeurs (Sun, IBM, HP, DEC, AT&T, Unisys, ICL, etc.) publie le standard X/Open Portability Guide Issue 3 (XPG3). Ce standard

s'occupe tout particulièrement des différences issues de la localisation géographique (date, alphabet, etc.).
Mais cela n'était pas encore suffisant, il fallait arriver à unifier les Unix pour qu'au bout du compte, une seule version s'impose au marché. Plusieurs efforts virent le jour dans ce sens en particulier le consortium OSF (Open Software Foundation). L'Open Software Foundation (OSF) fut une organisation fondée en 1988 en vue de créer un standard ouvert pour une implémentation du système d'exploitation Unix. Les entreprises fondatrices étaient les suivantes : Apollo Computer, Groupe Bull, Digital Equipment Corporation, Hewlett-Packard, IBM, Nixdorf Computer et Siemens AG. Les entreprises Philips et Hitachi les ont rejointes ultérieurement. L'OSF a fusionné avec l'X/Open en février 1996, pour donner naissance à l'Open Group.

Mais toutes ces initiatives n'ont pas donné grand chose finalement… Pire, l'incompatibilité grandissante entre les nombreuses variantes d'UNIX proposées par les différents éditeurs pour les différentes machines a fini par porter atteinte à la popularité d'UNIX. De nos jours, les systèmes UNIX propriétaires, longtemps majoritaires dans l'industrie et l'éducation, sont de moins en moins utilisés. En fin de compte, on peut dire que c'est GNU/Linux qui représente aujourd'hui "l'Unix unifié" rêvé par les promoteurs d'Unix d'hier…
Au sens strict, Linux n'est pas un Unix puisqu'il ne comprend pas de code provenant de l'original. D'ailleurs, Linux à proprement parler n'est que le noyau, le cœur du système d'exploitation. Le système d'exploitation est GNU/Linux, le noyau plus les outils basiques fournis par le projet GNU (pour Gnu's Not Unix : « Gnu N'est pas Unix ») de la Free Software Foundation.
Néanmoins, GNU/Linux a en commun avec Unix une bonne part de son fonctionnement et de son comportement. Pour un utilisateur lambda, il est bien difficile de faire la différence entre un Linux et un Unix.

A ce stade, nous avons passé en revue les grandes poussées du logiciel sur minis (Unix, L4G et SGBDR) ainsi que les ERP mais nous n’avons pas encore vraiment abordé l’affrontement majeur qui acheva de transformer le logiciel en marché de masse : la bureautique. Et là, c’est surtout la micro-informatique qui est concernée. Même si on l’a déjà évoqué dans l’évolution des L4G, le rôle des PC a été fondamental dans la transformation de la nature du logiciel. Et le PC nous a apporté deux choses : la bureautique et les interfaces graphiques (même si, sur ce dernier point, la contribution des stations de travail fut également important).

Nous évoquerons la bureautique un peu plus loin et voyons comment l'interface graphique a pu naitre et se développer dans le cadre de l'évolution de la micro-informatique...

==========
Encadré: l'évolution d'Unix

Unix est sans doute le système d'exploitation le plus adaptable de tous les temps. Originellement conçus pour les minis, Unix est présent sur tous les marchés (avec plus ou moins de succès), des plus gros supercalculateurs à l'informatique embarquée.

Bien évidemment, cette évolution ne s'est pas faite sans de multiples transformations. La première a été la montée des Unix commerciaux des vendeurs de machines Unix. Sun a eu SunOS puis Solaris, HP a eu HP-UX, IBM a eu AIX, Digital a eu Tru64 (tous dérivés soit de BSD, soit de System V, les deux principales branches "historiques d'Unix...), etc.

La seconde transformation s'est effectuée avec l'apparition de clones Unix sur

PC sous forme de logiciels libres (FreeBSD, Linux). Techniquement ces derniers ne sont pas des Unix, mais ils n'en gardent pas moins les caractéristiques de ce dernier. Cette transformation a été salutaire pour le monde Unix lorsque le PC s'est attaqué au marché des serveurs d'entreprise, reléguant au passage les machines Unix à un marché de plus en plus de niche. Ironiquement, Linux a été l'unification des Unix que tout le monde attendait, même si personne ne l'attendait sous cette forme.

L'aspect logiciel libre de Linux, couplé à la grande modularité d'Unix ont permit le passage à d'autres marchés comme les supercalculateurs et l'informatique mobile. Unix étant par nature très modulaire, il a par exemple pu perdre du poids pour s'adapter aux contraintes rigoureuses de l'informatique mobile. Mais l'énorme atout que Linux a fourni à Unix a été la possibilité de "voyager" sans demander aucune autorisation. La raison pour laquelle c'est Linux qui s'est imposé sur d'autres plates-formes et non Solaris, HP-UX et autres Tru64 est que Sun, HP ou Digital n'avaient aucun intérêt à porter leur système d'exploitation sur d'autres plates-formes, étant donné qu'ils vendaient du matériel. Au contraire, Google et les ingénieurs de Cray ont pu porter Linux sur respectivement un smartphone et sur le dernier supercalculateur sans avoir à demander la permission à Linux Torsvald.
==========

L'interface graphique : du projet de recherche à l'utilisation ordinaire

La lente évolution des interfaces graphiques prend sa source dans les années 60. A cette époque, les terminaux sont encore rares et une utilisation graphique de l'informatique n'est même pas envisagée. Pourtant, quelques pionniers vont ouvrir la voie et provoquer une explosion de pixels sur nos écrans.

Il faut se rappeler qu'avant l'écran, le principal mode de communication de l'ordinateur avec l'utilisateur était l'imprimante !

Sous Unix, l'éditeur de texte *ed* fonctionnait comme tel : on tape la ou les lignes que l'on veut afficher, et l'imprimante les affiche. On peut ensuite réécrire la ligne que l'on veut modifier. Bref, un travail bien fastidieux mais qui était optimisé pour les technologies de l'époque. A noter que *ed* est toujours fourni avec Unix, même s'il n'est plus utilisé.

En 1963, au MIT, Ivan Sutherland met au point le premier logiciel graphique interactif utilisant un stylo optique pour dessiner sur écran des schémas techniques. Ce programme (appelé "Sketchpad" ou même "Robot Draftsman") était vraiment révolutionnaire pour l'époque et a même fait l'objet d'un reportage à la télévision car les journalistes avaient l'impression de voir de la science fiction en direct !

Sketchpad était non seulement le tout premier logiciel de CAO mais aussi le tout premier programme à proposer une interface utilisateur "graphique" (quoique bien rudimentaire)

En 1967, le département informatique de l'université de l'Utah s'est spécialisé dans l'imagerie informatique en 3 dimensions. Ce département est alors dirigé par les professeurs David C. Evans et Ivan Sutherland qui fonderont la société Evans & Sutherland un peu plus tard.

La mère de toutes les démos !

Mais les vrais débuts démarrent vraiment en 68 quand Douglas C. Engelbart de la Stanford Research Institute (au département "Augmentation Research Center") fait une démonstration d'un environnement graphique avec des fenêtres à manipuler avec une souris.

Engelbart est vraiment une des figures marquantes de cette évolution. En 1962, il publia un article intitulé "Augmenting Human Intellect". Dans cet article fondateur, Engelbart explique que l'ordinateur pourrait être la façon la plus rapide d'améliorer les capacités humaines. Il y expose un environnement qui n'est pas destiné à remplacer le

cervaux humain mais plutôt un outil pour l'étendre. Et il donne des exemples comme cette hypothèse où un architecte conçoit un immeuble en utilisant un logiciel similaire aux programme de CAD qu'on connait aujourd'hui.
Il faut se représenter le saut intelectuel que représentait cet article en 1962... Les ordinateurs ne sont encore que des mainframes excutant des programmes en batch, l'utilisation même d'une console pour entrer des lignes de commandes est considéré comme la pointe du progrès !
Lors de cette fameuse démo de 1968, il démontre dans cet environnement l'utilisation d'un traitement de texte, d'un système hypertexte et d'un logiciel de travail collaboratif en groupe. Cette démonstration est un événement car tout y est : la notion de fenêtres, de périphériques dédiés (la souris bien sûr mais aussi le pointeur à l'écran de cette souris...), d'aide en ligne contextuelle, d'édition de documents, de messagerie interactive et même de vidoé conférence !
Jusqu'à ce jour, le noyau du système d'exploitation d'une mainframe IBM croit toujours qu'il mange des cartes perforées. Pour lui, un écran correspond à 25 cartes perforées pour les 25 lignes de texte (à vérifier). Autant dire que les mainframes ne sont pas adaptées pour le mode graphique, mais le sont beaucoup plus pour le HTML étant donné que ce dernier est du texte. Ce qui explique une certaine résurgence de ces dernières sur le Web.
La conférence d'Englebart était également une performance multimédia avec de multiple caméras de télévision pour montrer différents angles. Ce niveau de mise en scène était nécessaire car la plupart des concepts exposés étaient complétement neufs. Le système démontré était appelé NLS (pour *oN-Line System*) car il s'appuyait sur un plusieurs ordinateurs reliés entre eux. Tout cela paraissait venir de plusieurs décennies dans le futur et la majorité de l'assistance avait du mal à comprendre ce qu'elle voyait. Par exemple, le système NLS supportait le multi-fenêtres mais il n'y avait pas de bords bien délimités à ces fenêtres...

Le PARC prend le relais...
Au tout début des années 70, c'est Xerox qui prend le relais avec la création du centre de recherches PARC (Palo Alto Research Center) à Stanford. Le chercheurs du PARC travaillaient dans la plus grande liberté, Xerox ne leur ayant pas assigné d'objectifs commerciaux. Il faut noter que plusieurs personnes du Xerox Parc ont travaillé avec Douglas Engelbart auparavant.
Cette équipe de pointe va concevoir deux machines qui marquent les premières réalisations "industrielles" d'ordinateurs graphiques : l'Alto et la Star. En mars 1973, Le premier prototype de la station de travail Xerox Alto démarre pour la première fois.
L'Alto n'était pas vraiment un micro-ordinateur mais plutôt une station de travail avant la lettre : l'unité centrale tenait dans une tour qui pouvait être glissée sous le bureau. L'élément qui accrochait l'oeil immédiatement était l'écran : il avait une orientation verticale comme une feuille de papier et était capable d'adresser chaque point (pixel) avec une résolution de 606x808. Cette adressage de chaque point (le fameux "bitmapping") était tout à fait inhabituel pour l'époque car les terminaux ordinaires ne pouvaient qu'afficher des caractères à taille fixe. La souris était une version modernisée de la trouvaille d'Englebart et elle possédait trois boutons. Le pointeur de la souris était devenu une image qui, pour le première fois, prenait la forme d'une flêche inclinée qu'on connait bien désormais. le pointeur était également capable de changer de forme selon la tâche en cours (comme redimensionner une fenêtre).
Vue d'aujourd'hui, l'interface graphique (GUI pour *Graphical User Interface*) de l'Alto et du Star sont un curieux mélange de modernité et de rusticité : on a des fenêtres et les principaux widgets habituels (icones, boutons, ascenseurs) mais il manque les menus déroulants (introduit par Apple dans le Lisa) et la couleur !
Tous ces premiers systèmes graphiques sont systématiquement en noir et blanc, un paradoxe... Mais, à part ces réserves, force est de constater que 30 à 40 ans après, la seule chose qu'on est vraiment ajoutée à ces interfaces, c'est du relief et de la couleur. Soit ces premières réalisations représentaient un coup de maitre exceptionnel, soit on s'est endormi pendant des dizaines d'années et le réveil est forcément proche.

Le premier prototype opérationnel de l'Alto est terminé en Avril et c'est également à ce moment là que Dick Shoup met au point une machine dotée de la première carte graphique couleur capable d'afficher une image de 640x486 en 256 couleurs et aussi de numériser un signal vidéo.
Il réalise le programme Superpaint qui est à la fois un logiciel de dessin en couleurs et aussi le premier logiciel d'effets vidéo numériques. Comme ce projet était à l'opposé des objectifs de Xerox, il sera rapidement annulé. Dick Shoup démissionnera 2 ans plus tard et fondera sa propre société, Aurora Systems, qui commercialisera les premiers équipements permettant de génerer les logos et cartes météo numériques pour la télévision.
Avec quelques turbulences de ce genre, les recherches continuent au PARC et les premières s'accumulent : en février 1975, Charles Simonyi développe le tout premier traitement de texte WYSIWYG (What You See Is What You Get), le "Bravo".
Le premier logiciel écrit pour l'Alto était plutôt rustique et pas vraiment graphique. Le gestionnaire de fichiers affichait les répertoires en deux colonnes (un peu à la façon de Norton Commander) entourées par des boites mais il n'y avait pas de fenêtres comme nous les connaissons aujourd'hui. Le traitement de texte Bravo pouvait afficher différentes polices et tailles de caractères en même temps mais il avait sa propre interface utilisateur avec les menus en bas (le créateur de Bravo, Charles Simonyi, allait rejoindre Microsoft où il recréa le Bravo sous la forme du tout premier Word pour Dos...). Pareil pour Superpaint qui avait sa propre interface. L'équipe du PARC réalisa alors qu'elle avait besoin d'un environnement qui offre une interface utilisateur constante et cohérente pour les nouvelles applications, ce besoin entraina la création de Smalltalk, le tout premier GUI.
Quand on évoque ,Smalltalk, on pense tout de suite à l'aspect langage : "Smalltalk est un langage de programmation orienté objet, réflexif et dynamiquement typé, il est doté d'un environnement de développement graphique. Il a été conçu par Alan Kay, Dan Ingals, Ted Kaehler, Adele Goldberg et est inspiré par Lisp et Simula". Mais Smalltalk était plus que cela dans le cadre de l'Alto : non seulement il apporte le côté graphique au développement mais, en plus, c'est également l'environnement graphique dans lequel il s'exécute. C'était un peu comme si Microsoft avait développé Visual Studio comme une simple application qui assurait aussi le rôle de Windows ! Vous lanciez Smalltalk depuis le gestionnaire de fichier comme une application normale mais une fois en mémoire, il prenait le dessus et c'est par lui que l'environnement de travail de l'Alto passait (une sorte de "shell graphique"). Le concept d'icones apparut à cette époque : des petites représentations en images des fichiers sur lesquelles on clique pour les lancer ou les manipuler. Les "menus surgissants" viennent aussi de cette période : l'utilisateur cliquait sur un des boutons de la souris et des menus hiérarchiques basés sur la tâche en cours apparaissaient à la dernière position du pointeur de la souris. C'est également Smalltalk qui apporta tout le cortège de widgets auquel on est désormais habitué : boites de dialogues, radio-boutons, ascenseurs, etc.
La plupart des membres du PARC voulaient que Xerox commercialise la version III de l'Alto mais la direction de Xerox s'y refusa. Finalement, une version réduite de l'Alto fut quand même mise sur le marché, le Star 8010 Document Processor, fut proposé au public en 1981 pour $17,000. Le Star présentait quelques différences importantes avec l'Alto... La plus significative était que les fenêtres se juxtaposaient sur la Star alors qu'elles se recouvraient avec l'Alto. Mais le concept de fenêtres multiples empilées paraissait trop complexe et confus pour les gens du marketing de Xerox.
Xerox tenta de commercialiser la station Star (entre 1981 et 1986) mais ce fut un échec commercial que nous avons déjà relaté dans la section consacrée au matériel et aux constructeurs...

Mais l'interface graphique connait ses débuts en parallèle sur un marché tout autre : celui du grand public, à commencer par les jeux vidéos. En 1971, Nolan Bushnell crée en effet le premier jeu vidéo, Computer Space, suivi du célèbre Pong (1972) avant de créer la célèbre compagnie de jeu vidéos Atari. Pong est tellement populaire qu'un jour le propriétaire d'un bar où le jeu d'arcade était installé appelle Bushnell car le jeu ne

fonctionne plus. Affolé, ce dernier se précipite au bar, ses outils à la main, démonte la machine pour s'apercevoir que c'est l'afflux des pièces de monnaies qui a tout bloqué.

Les jeux vidéos vont grandement influencer la micro-informatique de l'époque, de l'Apple II (1978) à l'Amiga (1985). A l'époque, l'une des principales utilisations grand public des micro-ordinateurs était en effet les jeux vidéos. A l'époque, on jugeait d'ailleurs souvent les micro-ordinateurs par leurs capacités graphiques. Et si l'Apple II a donné naissance au tableur, un autre de ses logiciels célèbres était le jeu de rôle Wizardry (avec même une version française, Sorcellerie, inédit à l'époque). Steve Jobs a d'ailleurs travaillé chez Atari avant de fonder Apple.

Si l'informatique à cette époque-là n'est pas orientée interface graphique au sens moderne du terme, ses besoins en capacités graphiques ont mené à des ordinateurs qui ont eu les capacités d'avoir une interface graphique digne de ce nom. De par leur nature, les jeux vidéos ont toujours eu un grand besoin de puissance graphique, bien plus souvent que les applications d'entreprise (mis à part des marchés de niche comme la CAO). Les jeux vidéos vont donc engendrer un besoin en puissance graphique et donc un marché. Ce marché étant grand public, il a bénéficié d'énormes économies d'échelles. C'est la raison pour laquelle ce sont les jeux vidéos - et par extension la micro-informatique - qui ont fini par dominer complètement le marché du matériel graphique, plutôt que des constructeurs spécialisés de stations de travail graphiques tels que Silicon Graphics.

A partir de là, les choses s'accélèrent un peu car c'est aussi la période où la micro-informatique commence à tracer son sillon, profond et spectaculaire...
Apple est une des nouvelles vedettes de ce secteur et, en 1978, Apple Computer commence à travailler sur un projet de super-Micro Ordinateur. Nom de code de ce projet : Lisa.
L'orientation de Lisa prend un tour décisif quand un groupe de développeurs de chez Apple, dont Steve Jobs, assiste à une démonstration de l'Alto au sein du PARC (en décembre 1979).

La visite de Jobs au PARC laissa des traces : il fut si impressionné parce qu'il vit qu'il réorienta le projet Lisa afin qu'il ressemble au système Star... Mais cette imitation ne fut pas servile et apporta aussi son lot d'améliorations : le team Lisa innova dans la gestion des menus avec l'idée d'ajouter une cohce à droite de l'option sélectionnée ainsi les raccourcis-clavier pour les commandes les plus utilisées. Le team Lisa changea aussi certaines conventions comme le fait d'ajouter une corbeille pour supprimer des éléments ou de griser certaines options des menus quand celles-ci sont indisponibles (en fonction du contexte).
L'Alto avait une souris à 3 boutons et le Star avait une souris à 2 boutons pour plus de simplicité. Le team Lisa fit un pas de plus et introduisit le concept de souris avec un seul bouton. Mais comme l'interface demandait de séparer les actions possibles pour chaque icone (sélection et lancer par exemple), le "geste" du double-clic fut alors inventé pour apporter cette distinction. Le double-clic s'est ensuite standardisé dans tous les cas, même pour les systèmes avec des souris à plusieurs boutons...
Le geste de "drap & drop" (glisser-déplacer ou glisser-lâcher) vient également de cette équipe qui affina le GUI de l'Alto au point où toutes les interfaces graphiques que nous utilisons aujourd'hui lui ressemble encore !

Le Lisa fut mis sur le marché en 1983 mais ne connu pas plus de succès (100000 exemplaires vendus), toujours à cause d'un prix de vente rédibitoire. Mais le projet Macintosh était là pour prendre le relais. Le mac était destiné à offrir plus ou moins la même chose que le Lisa mais pour 4 fois moins cher !
Le Mac était en quelque sorte, la deuxième génération des Lisa, et Apple avait appris de cette expérience et éliminé de nombreux bugs. L'interface utilisateur du Mac était très étroitement dérivée de celle du Lisa et elle partageait même une partie du code mais le système d'exploitation du Mac avait été entièrement réécrit afin de loger dans

un espace mémoire plus réduit...
Les premiers systèmes bénéficiaient ainsi beaucoup de leurs prédécesseurs : le Star s'appuyait sur l'expérience accumulée avec l'Alto, le Lisa ne débutait pas de zéro grâce à la Star qui montrait déjà un concept apuré et le Mac gommait les imperfections (nombreuse) du Lisa...

Pendant ce temps, le monde du PC constate que ces nouveaux produits proposés par Apple sont très en avance sur ce que MS-Dos est capable de proposer... Microsoft promet donc formellement (en novembre 83) que son interface graphique pour l'IBM PC sortira en Avril 1984. Il faut dire que Bill Gates a lui aussi été particulièrement impressionné par ce que propose Xerox au point d'investir $100 000 pour acheter un système Star pour lui tout seul...
De son côté, Digital Research, vexé d'avoir raté l'opportunité MS-Dos commercialise son interface graphique GEM pour IBM PC dès Septembre 1984 (espérant ainsi prendre sa revanche sur Microsoft en investissant très tôt un domaine d'avenir). Microsoft présente en avril 84 "Interface Manager" (renommé par la suite Windows), un concept d'interface graphique pour le PC, et annonce sa sortie prochaine...
On a du mal a se représenter l'effervescence qui régnait à cette époque à ce sujet : tous les éditeurs importants du monde du PC voulaient prendre pied sur ce nouveau domaine tellement la démonstration faite par Apple semblait indiquer la voie à suivre. Microsoft, Digital Research et Visicorp s'affrontaient à coup d'annonces et de travail forcéné (même si les résultats concrets tardaient à jour le jour).
L'histoire de Visi On de l'éditeur de Visicalc est sur ce point particulièrement significative : avec un développement sur VAX commencé dès l'automne 81, l'éditeur commit l'imprudence de montrer une version beta lors du Comdex 82... Le succès relatif de cette démonstration impressionnante (Gates inquiet du battage autour de Visi On prétendit que la démo ne s'exécutait pas sur un PC mais sur un terminal relié à un VAX...) poussa l'éditeur à commercialiser dès décembre 83 un suite d'applications reposant sur ce système pour plus de $1700 !
Le multi-tâches était l'argument principal de Visi On en plus de l'aspect graphique mais, à cette époque, on utilisait un PC principalement pour une application, parfois deux et donc cet "avantage" était un peu trop en avance. De plus Visi On réclamait aussi de l'espace sur un disque dur, le coût total montait presque au deux-tiers du prix du Lisa... Et tout cela pour un environnement qui fonctionnait à peine !
Car Visi On était terriblement lent et la sortie du Macintosh peu après montrait qu'on pouvait faire un système comparable pour moins cher, plus rapide (à peine...) et surtout plus beau (l'interface du Mac était élégante et l'écran lui rendait bien alors que Visi On sur un écran de PC donnait envie de pleurer tellement c'était triste en comparaison !). Visi On utilisait le mode d'affichage "graphique" CGA du PC en monochrome (640x200). Visicorp tenta de relancer Visi On plusieurs fois en baissant le prix de l'ensemble mais le produit était condamné et Visicorp ne résista pas à cette débacle.

Windows 1.0 est enfin dévoilé lors du salon Comdex 85 et Microsoft annonce sa vente pour juin de la même année au prix de $95. L'éditeur de Redmond met finalement Microsoft Windows 1.0 sur le marché en novembre 85, soit deux ans après son annonce, au prix de $100. Les toutes premières photos d'écrans ressemblent à un croisement entre Visi On et la version 1.0 de Word pour Dos.
La toute première version de Windows affiche même le concept rétrograde des fenêtres juxtaposées (les fenêtres de Windows 1.0 ne peuvent pas se recouvrir les unes les autres). On peut penser que cette limitation volontaire de Microsoft était là pour empêcher la prolifération des fenêtres et ainsi éviter au système de planter trop vite... La première version sera tout de même en couleur et aura tous les attribus habituels des GUIs comme les ascenseurs, les menus et les widgets. Une autre différence notable par rapport à l'interface du Lisa : chaque application a son propre menu attaché à sa fenêtre, juste sous la barre de titre.

Tous ces développement (en plus de GEM et de Windows, il faut aussi évoquer Visi-On)

ont eu des gestations pénibles et les premières versions font peine à voir : non seulement c'est moche mais en plus, c'est très lent !
Cela s'explique facilement : les PC de l'époque ne sont ni conçus ni capables de faire tourner autre chose que des applications en mode caractères... Les processeurs sont trop justes (le 8086 d'Intel est loin des performances du 68000 de Motorola... cela changera avec le 80386 mais celui-ci n'apparaitra que bien plus tard) et surtout, les cartes vidéo ne sont pas prévues pour faire du bitmapping, caractèristique essentielle des stations de Xerox ou des machines d'Apple (Lisa et Mac). Sans bitmapping, votre affichage reste grossier et votre interface ressemble à un maquillage raté (en plus d'être lente !).
Etouffés sous le poids de ces interfaces graphiques, le résultat est pitoyable et les ventes sont évidemment ridicules.

Jusqu'au début des années 90 ce sont les micro-ordinateurs non compatibles PC qui vont tenir le haut du pavé en terme d'interface graphique. Le Macintosh bien sûr (1984), mais également l'Atari ST (1985) - surnommé le Jackintosh, du nom du PDG d'Atari Jack Tramiel - et l'Amiga (1985), connu pour son processeur graphique révolutionnaire pour l'époque. Ces trois ordinateurs utilisant un Motorola 68000 et étant bien mieux conçus que le PC, ils auront dés le début un environnement graphique standard (fenêtre, souris, etc.) et se doteront rapidement de traitements de texte et de logiciels de PAO utilisant des fontes proportionnelles. Certains d'entre eux réussiront à imprimer des documents d'une qualité remarquable en utilisant une simple imprimante matricielle - et ce en utilisant jusqu'à 8 passes par ligne. Le système d'exploitation du Macintosh a d'ailleurs eu des fontes proportionnelles en standard dés le début en 1984, contrairement aux autres ordinateurs qui utilisaient de fontes 8x8 pixels, même en mode graphique (y compris l'Atari ST et l'Amiga). L'Amiga, quant à lui, a eu avant tout le monde un processeur graphique spécialisé lui permettant de définir ses modes graphiques (résolution, nombre de couleurs) à sa guise, de la même manière que les cartes graphiques le permettent à l'heure actuelle (à l'époque les autres ordinateurs avaient quelques modes graphiques prédéfinis). Le PC mettra des années avant de rattrapper ces ordinateurs. Ce n'est qu'à partir de Windows 3.0 (1990) que le PC aura des fontes proportionnelles. A noter que la raison de l'avancée du Macintosh est encore une fois due à Steve Jobs qui a pris des cours de calligraphie à l'université. Steve a reconnu que s'il n'avait pas pris ces cours, le Macintosh n'aurait pas eu de fontes proportionnelles.

Windows 2.0
Windows revient dans une seconde version un peu moins risible à la fin de 1987. La version 2.0 adopta à son tour le principe des fenêtres superposées et profitait de la puissance croissante des PC de l'époque pour s'exécuter à peu près normallement. Dans la foulée (en 89) de cette version 2.0, HP sortit une extension un peu plus élaborée appelée New Wave qui poussa Apple a intenter un procès à HP et Microsoft pour imitation du "look & feel" du Mac... Proècs qui fut finalement perdu par Apple.
Il est important de noter que les GUIs de cette époque (milieu et fin des années 80) utilisent tous des polices de caractères à taille fixe pour l'affichage système. Dans les applications, les fontes proportionnelles étaient supportées mais pour tout ce qui était système (labels des menus, étiquettes des icones), on avait systématiquement recours à des polices fixes par soucis de clarté. C'est aussi à cause des limites du matériel que cette contrainte restait vivace. Même le Mac avec une résolution verticale de 384 (bien supérieur aux 200 pixels communs aux PC) avait une police spécialement conçue (Chicago) pour cette affichage parce qu'elle était facile à lire, même quand les options étaient grisées. Avec l'évolution du matériel, les fontes proportionnelles se sont généralisés, même pour l'affichage système.

Le monde des stations de travail n'est pas en reste car là, la puissance disponible est suffisante... En 1983, Le MIT commence à développer le X Window System, un logiciel permettant de gérer l'affichage graphique des stations de travail Unix. Plus qu'une simple interface graphique, il s'agit d'un système client-serveur évolué capable par

exemple de gérer plusieurs écrans sur une même machine ou d'afficher sur l'écran d'une machine distante. Le MIT publie la première version de son environnement graphique pour station Unix en 86, il s'agit de X v10.4.
Dans l'environnement des stations de travail, le mode graphique était déjà répandu puisque Sun, par exemple, proposait NeWS (Networked Windows Sytem). Mais X-Window proposait trois avantages majeurs sur les environnements propriétaires comme NeWS :
1- X était indépendant du matériel (principe de d'indépendance vis-à-vis du matériel). En d'autres termes, il pouvait tourner sur n'importe quelle station de travail du marché ce qui était un plus énorme à une époque où la diversité était la norme.
2- X proposait un mode de répartition des traitements reposant sur TCP-IP (principe de distribution des traitements sur le réseau). Avec X-Window, l'application était en fait cliente d'un serveur d'affichage (qui pouvait être un simple terminal du moment que ce dernier était conforme à la norme X-Window ce qui fait qu'on appelait ces dispositifs des "terminaux-X"...).
3- X séparait l'affichage effectif des mécanismes (principe de séparation des couches) ce qui fait que des gestionnaires de fenêtres spécifiques pouvaient être employés en haut de la pile (le plus connu de tous était MOTIF).
En fait, la norme X-Window représentait la première vraie avancée significative depuis l'Alto : le Mac avait amélioré l'interface utilisateur imaginée au PARC mais il restait un système monolithique et propriétaire. X, au contraire, était non-seulement ouvert mais incluait un fonctionnement distribué en natif !
Ceci dit, X avait aussi les défauts de ces qualités : le principe de séparation avait pour conséquence une prolifération des toolkit et des window manager... Le window manager gérait la création et la manipulation des fenêtres et des objets à l'intérieur de ces fenêtres mais ce n'était pas pour autant une interface graphique complète. Une autre couche logicielle était nécessaire au sommet de la pile : l'environnement de bureau ("desktop environment" ou DE). Cette dernière couche dépendant de chaque vendeur et, du coup, le "look & feel" de l'interface de Sun était différente de celle de SGI.
Le standard X était ainsi très morcelé au final ce qui n'était pas trop gênant dans le monde des stations de travail qui s'adressait à des ingénieurs (et qui donc pouvaient passer un peu de temps et d'énergie à s'y retrouver dans ce chaos, faire leur choix et arriver à l'installer...) était éliminatoire dans le monde de la micro-informatique nettement plus orienté "tout public"...

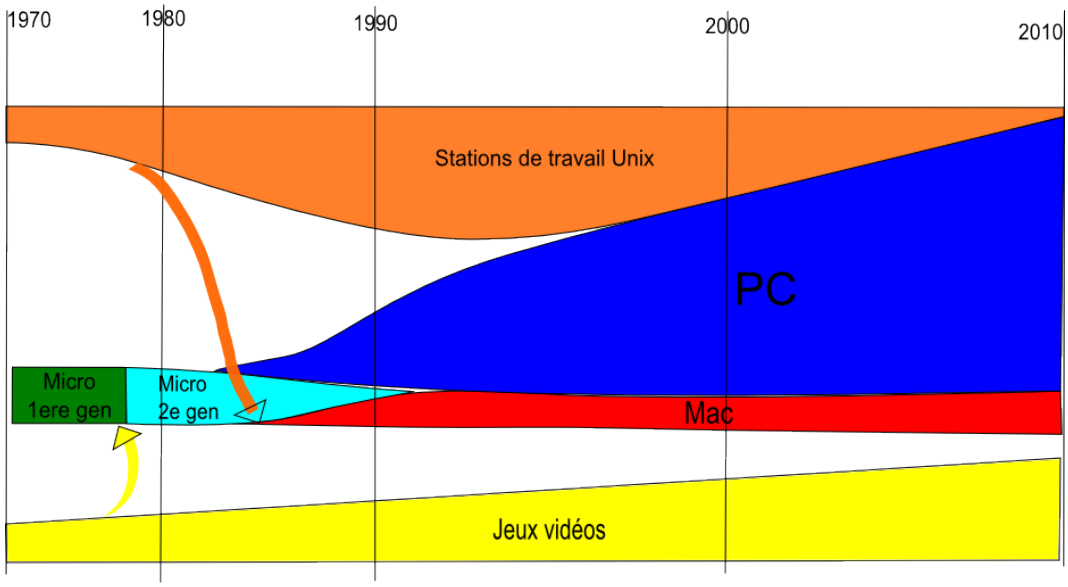
Entretemps, Steve Jobs s'est fait virer d'Apple et, avec 5 ex-dirigeants d'Apple, ils fondent NeXT Incorporated pour développer un "meilleur Macintosh". L'interface graphique du premier "Cube" de Next va représenter un nouveau pas en avant en 1988 (malheureusement en noir & blanc !). NeXTSTEP (c'était le nom du GUI du "cube" Jobs) avait une apparence 3D élégante qui tranchait avec la 2D habituelle de Windows et du Mac. C'est avec qu'on commence à voir un petit "x" au coin de la fenêtre et qui permet de fermer cette dernière. NeXTSTEP avait aussi un "dock" qu'on pouvait mettre d'un côté ou l'autre de l'écran.

OS/2 et Windows 2 puis 3, le début de la normalisation
C'est également en 1988 qu'arrive la première version graphique d'OS/2 mais Presentation Manager (c'est le nom du GUI d'OS/2) est presque exactement identique à Windows 2.0. Au même moment, Microsoft propose une évolution de Windows 2, la version 2.1. Cette version 2.1 n'apporte rien de nouveau sur le plan visuel mais est proposée en deux déclinaison : Windows/286 et Windows/386. Ces deux déclinaisons étaient capables (surtout la seconde) de dépasser la limite de 640 Ko de mémoire du Dos et donc de fonctionner avec beaucoup plus de confort. Cette "perçée" annonçait clairement l'évolution à venir... Windows 3.0 sortit en mai 1990 et ressemblait un peu à New Wave... Et surtout, il supportait le mode d'affichage VGA en 256 couleurs (en plus d'être capable d'adresser toute la mémoire disponible sur chaque PC), un grand progrès par rapport aux versions précédentes !

A partir de là, l'interface graphique s'est vite généralisée pour tous les usages et pas seulement dans le cadre des applications dites de "productivité individuelle", leur terrain d'origine. En effet, on s'est aperçu que l'interface graphique n'était pas seulement utile pour les tâches bureautiques, son bénéfice était également important pour toutes les applications de gestion réclamant beaucoup de saisie. Si dans vos applications de production vous utilisiez le mode graphique et tous ses avantages (case à cocher pour indiquer la présence d'un élément, bouton pour sélectionner un choix parmi trois ou quatre, valeurs apparaissant dans une petite fenêtre sur laquelle on clique pour désigner une valeur dans une liste…), vos utilisateurs pouvaient alors aller jusqu'à quatre fois plus vite en faisant moins d'erreurs de frappe et de fautes d'orthographe qu'avec de classiques transactions en mode texte. Plusieurs études, datant des années quatre-vingts, comparant la productivité et la courbe d'apprentissage d'utilisateurs d'interfaces graphiques par rapport à des utilisateurs d'interfaces caractères ont démontré des différences allant jusqu'à 200 % en faveur des interfaces graphiques.
Depuis le milieu des années quatre-vingt jusqu'au milieu des années quatre-vingt-dix, le domaine de l'ergonomie des interfaces utilisateurs a été dominé par les interfaces graphiques (GUI pour Graphical User Interface). Les concepteurs qui ont défini les règles régissant les premiers GUI cherchaient une alternative acceptable aux interfaces de type ligne de commande, comme MS-Dos, avec des représentations graphiques du système de gestion de fichiers local. Leur but était de rendre facile à gérer quelques Mo de ressources locales.
C'est la finesse de cette gestion événementielle qui fait toute la richesse des applications sous Windows. Mais, la gamme des possibilités théoriquement illimitées de Windows (et des autres GUIs) peut constituer un sérieux handicap dans de nombreux cas. À titre d'exemple, il peut devenir gênant d'avoir à choisir entre les nombreuses façons d'accomplir une même fonction. De fait, il existe toujours plusieurs « gestes » pour une même tâche avec une interface graphique. Pour un contexte de type application de gestion, ce potentiel très large constitue un inconvénient plutôt qu'un avantage, les nombreuses options se traduisant souvent par autant d'occasions de se tromper.
Par analogie, on peut dire que réaliser une interface utilisateur sous Windows revient à développer en natif : on n'est pas limité par l'environnement mais il n'est pas aisé de faire le tri entre les choix multiples et trouver les bonnes fonctions à activer n'est pas simple. La « largeur de bande » de Windows a pour conséquence que le « signal » est beaucoup plus dur à focaliser sur tel ou tel domaine d'application, et en particulier pour les applications de gestion.
Les développeurs ont, pour la plupart, souffert de ce choc culturel. Mais ce ne sont pas les seuls !
Les utilisateurs aussi ont eu leur part de difficultés face à une interface riche de possibilités… Avec le recul, on sait aujourd'hui qu'on doit reconsidérer en profondeur certaines caractéristiques qui posent beaucoup de problèmes à la grande masse des utilisateurs comme le double-clic (MS Word a même été jusqu'à proposer le triple-clic pour permettre de sélectionner un paragraphe de texte...), les fenêtres qui se recouvrent (cet avatar du multi-tâches n'est pas si évident à assimiler pour les utilisateurs occasionnels et c'est pourquoi l'iPad d'Apple est limité à une seule application, un seul mode à la fois), les menus hiérarchiques en cascade et des barres d'outils qui ressemblent à des hiéroglyphes.
On s'est également rendu compte que la plupart des utilisateurs en entreprise n'avaient rien du gestionnaire de la connaissance tant vanté par les brochures marketing. Ils aspirent plutôt à exécuter les tâches qui leur sont dévolues aussi simplement que possible. Ils peuvent donc mettre à profit un ensemble de possibilités réduit et facile à maîtriser, tout comme le photographe occasionnel se contentera d'un appareil simple et ne saura que faire d'un modèle trop sophistiqué.

C'est dans ce contexte que le Web est apparu comme une alternative bienvenue pour passer outre les limites de la trop grande richesse des GUIs traditionnelles.

==========
Encadré: l'évolution des marchés de l'interface graphique



[essai de graphique - les tailles de marché ainsi que la chronologie ne sont pas respectés. Je ne sais pas si c'est suffisamment clair pour être utilisable, et si cette section est au bon endroit]

Ci-dessus le graphe des machines qui ont supporté les différentes interfaces graphiques du marché. Si la micro-informatique (par le PC) a été amenée à dominer le marché, elle n'en n'a pas moins subi l'influence de deux autres marchés. Les jeux vidéos tout d'abord ont influencé la seconde génération (les premiers micro-ordinateurs comme produit finis: Apple II, Commodore 64, Sinclair, etc.), ce qui a à terme amené cette dernière à booster ses capacités graphiques. La seconde influence est venue des stations de travail Unix qui, ayant plus de puissance que les micros, ont pu expérimenter les premières interfaces graphiques - mais l'idée a finalement voyagé jusqu'aux micros. Ces deux influences ont conduit au Macintosh puis à Windows. Ironie du sort, la micro-informatique est revenue vers le marché des stations de travail Unix qu'il a écrasé - ces dernières ne sont plus qu'un marché de niche. Du côté des jeux vidéos, il existe une certaine concurrence entre PC et consoles de jeux vidéos, même si elle n'a pas été aussi exacerbée qu'entre le PC et les machines Unix.
==========

Le Web, changement de direction ou retour en arrière ?
Le Web a bien évidemment changé beaucoup de choses en terme d'interface graphique. Comme toute technologie disruptive, il a au départ apporté des interfaces moins riches, surtout à ses débuts (les vrais débuts du Web sont à situer en 1994/ 1995). Un des changement a été la manière de présenter les données: sous la forme d'une page à hauteur aussi grande que nécessaire. Mais le changement le plus important a été la saisie: au lieu de saisir tous les champs dans une même fenêtre (ou page), les premières applications HTML nécessitaient de changer de page dès que certains champs étaient liés. Par exemple, un premier écran demande d'abord la région géographique désirée, et un second affiche les sous-régions en fonction du premier choix. Cette conception avait une certaine lourdeur, mais a fonctionné.

En parallèle du développement de HTML, de nombreuses technologies (toutes propriétaires) ont voulu compléter et/ou remplacer ce dernier, et en particulier permettre d'avoir des pages plus riches que ce que le HTML proposait. Plusieurs technologies se sont donc développées pour lancer un programme au sein d'une page HTML. Les deux technologies les plus célèbre du moment ont été ActiveX de Microsoft, qui embarquait un programme Windows à part entière au sein d'Internet Explorer, et Java de Sun. Mais aucune de ses deux technologies n'ont vraiment pris à part pour des besoins les plus extrêmes comme WebEx, un service de téléconférence / accès à distance.
Java n'était pas appelé à reproduire l'interface graphique à la place de l'interface du Web parce que justement, l'interface graphique n'est PAS ce qui convient pour obtenir des applications intuitives et simples à utiliser. L'interface hypertexte du Web a prouvé qu'elle offrait une solution qui plaisait à la grande masse des utilisateurs et qu'elle permettait de mettre en place de véritables applications (avec des limites, certes mais acceptables en regard de la simplicité qu'elle apportait).
La seule technologie qui a su se faire une place au soleil a été Flash de Macromedia (racheté par Adobe en 2004), une plateforme fortement orientée graphisme et animation - contrairement à ActiveX ou Java qui étaient des langages de programmation traditionnels adaptés pour tourner au sein d'un navigateur Web. Flash a en particulier eu énormément de succès avec la montée des service de vidéo en ligne tels que YouTube. Si Microsoft Media Player a eu début de succès pendant un temps, les vidéos à base de Flash l'ont finalement supplanté car beaucoup plus robustes. Microsoft a depuis développé Silverlight pour concurrencer Flash, pour l'instant sans succès. Flash reste populaire pour les vidéos en ligne, les publicités interactives et certains services qui désirent offrir des effets. Un traitement de texte à base de Flash a été développé (BuzzWord), mais pour l'instant n'a pas eu grand succès.

Mais comme toute technologie disruptive, le HTML s'est amélioré. Peut-être pas au point de rattrapper les interfaces graphiques riches (et "lourdes" aussi...), mais suffisamment pour être de plus en plus efficace. Il s'est doté d'un langage de script avec JavaScript (qui n'a de commun avec Java que le nom), et chaque version successive de HTML a apporté de plus en plus de fonctionnalités. Grâce à la technologie AJAX (Asynchronous JavaScript and XML) les applications Web ont pu mettre à jour certaines parties de l'écran sans avoir à rafraichir la page entière. La première application célèbre utilisant AJAX a été Gmail, le service en ligne d'email de Google. Gmail a innové par son interface novatrice, regroupant les divers messages d'une même conversation en un seul écran, et permettant d'extraire ou de contracter les messages désirés afin de gagner de la place. Google a par la suite repoussé les limites encore plus loin avec Google Maps en permettant à l'utilisateur de faire glisser la carte à la souris. Les précurseurs de Google Maps tels que Mapquest ou Yahoo Maps demandaient de cliquer sur un côté de la carte pour se déplacer, ce qui entrainait un rafraichissement complet de la page. Google Maps a tellement ébahit tout le monde (on dit que même Bill Gates a été surpris par ce que le service pouvait faire) que le service de Google a détroné Mapquest (le service dominant du domaine jusque-là) presque du jour au lendemain.
A tel point que HTML menace de plus en plus Flash. Autant ce dernier n'a plus que des amélioration mineures à apporter en terme d'expérience utilisateur, autant HTML essaie de s'améliorer à chaque version. Apple a déjà shunté Flash sur sa plate-forme mobile iPhone / iPod Youch / iPad. Et HTML 5 permettant de jouer des vidéos à la place de Flash, il existe des plug-in qui désactivent Flash complètement et remplace ce dernier par une vidéo HTML 5 sous YouTube.

Maturité
En dépit de l'apparente homogénéisation des interfaces logicielles, il y avait encore un peu de place pour l'innovation. Windows 95 a signalé un abandon de l'interface de Windows 3.x (où les principales fonctions étaient regroupées dans la fenêtre du Program Manager) au profit d'une interface plus proche du Macintosh (où les principales fonctions proviennent d'un menu principal ainsi que d'icônes personnalisées

sur le bureau). Windows 95 introduit ainsi le concept de menu "start" (dont on se servait aussi pour éteindre la machine...) où tous les programmes pouvaient être lançés et de la barre de tâches où on pouvait passer d'une application à l'autre. A noter que Microsoft a manqué de quelques pixels de reproduire une fonctionnalité du Mac: sur ce dernier, il suffit en effet de faire remonter la souris tout en haut de l'écran et de cliquer pour avoir le menu. Verticalement parlant, pas besoin d'avoir à positionner la souris avec précision, on peu "écraser" le curseur tout en haut, cliquer et on accède au menu. Sur Windows 95, par contre, en "écrasant" le curseur en bas à droite on loupait de quelques pixels le bouton Démarrer... De son côté, Apple ne reste pas immobile puisque le nouveau système d'exploitation pour le Macintosh est issu de la fusion avec NeXT et on peut considérer qu'il s'agit d'une énième version de NeXTSTEP. C'est l'occasion d'une refonte et d'un raffinage de l'interface utilisateur avec un soin du détail qui dénote une arrivée à maturité, plus de 30 ans après l'Alto et Smalltalk !

L'informatique mobile, encore une autre voie...
Dés les premiers PDA, l'informatique mobile a été confronté à un problème de taille (au propre comme au figuré): un appareil *beaucoup* plus petit qu'un ordinateur. L'écran est non seulement beaucoup plus réduit, mais le clavier est également minuscule (quand il y en a un). Qui plus est, il n'y a pas vraiment la place pour un trackpad, et bien évidemment pas question d'utiliser une souris. Il a donc fallu innover pour faire face à ce challenge, et la technologie centrale a été et reste les écrans tactiles du fait de leur gain en place. La première tentative avec l'Apple Newton s'est soldée par un échec, mais le Newton a tracé la voie, car tous ses successeurs, du premier PalmPilot aux plus récents iPhone ont tous des écrans tactiles.

L'utilisation d'un écran tactile s'est-elle améliorée. De l'utilisation d'un alphabet simplifié pour éviter les fautes (PalmPilot) on est passé à un écran virtuel (iPhone). De même, les smartphones ont commencé à être multi-tactiles, permettant à l'utilisateur de "pincer" une image pour l'agrandir ou la rétrécir (encore une innovation de l'iPhone qui a su jouer à fond sur "l'effet démo" !).

L'accéléromètre a fait partie des technologies utilisées. La encore, il semble que l'origine provienne des jeux vidéos. En 1989, Nintendo a sorti le PowerGlove, un gant en plastique muni d'un accéléromètre. Si le PowerGlove a été un échec, il a tracé la voie de la Nintendo Wii (2006) qui a eu un énorme succès du fait de ses contrôleurs novateurs. Et en 2007, l'iPhone créait la sensation avec l'utilisation d'un accéléromètre pour savoir si l'appareil était disposé à l'horizontale ou à la verticale. Mais la principale utilisation de l'accéléromètre pour l'iPhone a été pour les jeux.

L'histoire du développement et de l'évolution des interfaces graphiques est longue et compliquée. Même s'il est facile d'identifier quelques indivudus clés comme Douglas Engelbart et Alan Kay qui ont fait de grandes et décisives contributions, la vérité est que l'interface graphique est le fruit d'un travail continuel de nombreuse personnes pendant une longue période (plus de vingt ans, presque trente !). Dire que c'est Apple qui a inventé ce type d'interface ou que Jobs en a volé l'idée à Xerox est trop simpliste. Il est également inexact d'attribuer le seul mérite à l'équipe du PARC de Xerox (même si c'est bien là que les plus grands progrès ont été réalisés). En fait, les différentes équipes qui se sont succédés dans cette aventure ont toutes empruntées les unes des autres pour améliorer et raffiner le concept progressivement.
Il est frappant de constater que les interfaces des PC d'aujourd'hui semble stagner (le relais parait être passé aux mobiles et en particulier aux smartphones... Le touchscreen de l'iPhone représente une vraie perçée en la matière) alors que, dans le même temps, les développeurs d'applications éprouvent toujours les mêmes difficultés à respecter les principes fondateurs des GUIs qui, pourtant, accusent désormais une certaine ancienneté...

La bataille de la bureautique

L'évolution de la micro-informatique a surtout été marquée par une guerre technique et commerciale sans répit entre les différents éditeurs, chacun spécialisé sur sa niche de marché, (au milieu des années 80, le marché se répartissait principalement entre Lotus sur les tableurs, Wordperfect sur les traitements de texte, Asthon Tate avec dBase III sur les bases de données, Borland sur les langages et ainsi de suite, j'en oublie forcément) jusqu'à ce que Microsoft mette tout le monde d'accord avec sa suite Office !

Le marché du logiciel se structure grâce à l'effet de volume qu'apporte le PC
Au début du décollage des logiciels, il ne s'agit pas encore d'un marché très structuré. Après les SSII qui tentaient de commercialiser des logiciels plus ou moins à la suite d'un projet réussit chez un client, sont apparus les vrais développeurs qui se lançaient sur le marché avec une idée en tête et l'envie de diffuser leur produit.
Mais la société typique de cette première époque, c'est habituellement un ou deux développeurs, bon techniciens mais pas forcément calés sur l'aspect business de ce secteur. C'est donc tout naturellement que sont apparus sur le marché des intermédiaires spécialisés qui eux avaient pour vocation de faire la liaison entre les créateurs de programmes et les utilisateurs/clients de ces logiciels.
Cette chaine de distribution s'est surtout matérialisée avec la croissance du PC qui entraina la diffusion massive des logiciels phares dans chaque catégorie. On l'a déjà évoqué mais on n'avait jamais encore vu de pareils volumes dans le monde de l'informatique qui, jusqu'au début des années 80, restait un microcosme assez étroit (on est passé en une décennie à quelques dizaines de milliers de machines -des minis et des mainframes- à quelques millions -surtout des pc).
Dans cette chaine, c'est le distributeur qui s'occupait de répartir les packages chez les détaillants (points de vente physiques ou par téléphone). Ce rôle de distributeur pouvait être assimilé à celui du grossiste dans d'autres secteurs.
A l'age d'Or du PC (durant la période 1981/83, les revenus des producteurs de logiciels pour micro passèrent de 70 à 486 millions de dollars… ça donne une idée de ce qui s'est passé pendant cette quelques années !), le plus fameux distributeur fut sans conteste SoftSel, créé en 1980 en Californie. Trois ans plus tard, il employait 340 personnes et affichait un chiffre d'affaires de $80 millions. En plus d'avoir plus de 3000 références à son catalogue de l'époque (être dans son catalogue était indispensable si on voulait "exister" sur le marché…), SoftSel publiait une "hot-list" périodique qui devint vite le hit-parade de référence de la profession.
En amont, entre le développeur et le distributeur, on trouvait l'éditeur. On pense habituellement que l'éditeur et le développeur sont une seule et même entité et c'est souvent vrai mais pas toujours. Dans le monde du jeu vidéo par exemple, la différence est souvent nette entre les sociétés qui éditent les jeux et celles qui les développent (les éditeurs comme Electronics Arts sont connus du grand public parce que leurs noms apparaissent en gros sur les boites des jeux alors que ceux des studios de développement sont nettement plus discrets). Les premières s'occupant de traduire, documenter et de packager le logiciel mis au point par les secondes…
Dans le domaine du logiciel bureautique, cette répartition des rôles est aussi arrivée dans quelques cas illustres. En particulier avec Visicalc développé par Software Arts et édité par Personal Software. Ce fut également déterminant dans le cas de dBase écrit par Wayne Ratcliffe (qui avait d'abord appelé son produit "Vulcain") qui n'aurait sans doute pas fait la carrière que l'on sait si l'éditeur Ashton Tate ne s'en était pas occupé…

Très vite, c'est le marketing qui prit plus d'importance que la technique dans le secteur en effervescence du logiciel pour PC et c'est Lotus qui donna le ton le premier. Pour lancer son tableur 123, Mitch Kapor, jeune dirigeant de Lotus Corp. dépensa en publicité (y compris pour des spots tv, une première à l'époque) 2,5 des 4 millions de dollars qu'il avait réussi à lever auprès des professionnels du capital risque (Seven-Rosen principalement)… Un pari audacieux mais un pari gagnant : 123 s'est très vite imposé comme LA référence dans le crénau des tableurs, loin devant Visical, Supercalc et Multiplan (il faut dire aussi que 123 présentait des qualités techniques intéressantes dues aux choix judicieux opérés par Kapor qui avait une vision juste de ce que le marché attendait à cette époque).

Donc, lors de la première vague des éditeurs majeurs du PC, les leaders sont Lotus, WordPerfect et Asthon Tate... Où est donc Microsoft ?

Microsoft ou la prise de pouvoir par MS-Dos

On a souvent dit que c'est MS-Dos qui fit la fortune de Microsoft et c'est vrai : jusqu'en 1990 (au moment de la sortie de Windows 3.0 donc), MS-Dos représentait encore 20% du chiffre d'affaires de l'éditeur. Mais, à partir de là, c'est aussi les logiciels applicatifs qui ont participé à la croissance de la firme de Redmond (avec Office comme nouvelle "vache à lait" au côté de Windows qui replace progressivement Dos à partir de là).
On a également souvent affirmé que c'est IBM qui offrit cette place en or à Microsoft et là, ce n'est pas tout à fait exact. IBM a certes été assez maladroit de laisser Microsoft développer son OS à sa place (mais le contexte du projet au sein de big blue était "faites vite" et pour cela "n'hésitez pas à externaliser" ce que l'équipe de projet appliqua à la lettre… utilisez un operating system développé au sein du géant aurait pris des années !) mais Microsoft a sa part dans cette conquête de ce marché clé.
On sait que Digital Research débuta mal les choses avec les représentants d'IBM mais ce n'est pas pour autant que l'éditeur de CP/M s'est retrouvé totalement hors course par la suite… En effet, à la sortie du PC, IBM mettait en avant 3 systèmes d'exploitation et donc pas seulement MS-Dos (appelé PC-Dos quand il était commercialisé par Big Blue) mais aussi UCSD p-Systme de Softech et CP/M86 de Digital Research (CP/M86 était un portage de CP/M pour le processeur 8086 d'Intel alors qu'il était conçu au départ pour le Z80 de Zilog). UCSD était très lent et ne pesa donc pas lourd dans la balance… Restait MS-Dos et CP/M86… Mais ce dernier ne fut pas disponible tout de suite, un retard qui détermina son destin : les sociétés qui voulait commercialiser des logiciels pour le PC d'IBM voyaient bien que le choix se résumait à MS-Dos si elles voulaient être sur le marché tout de suite.
Digital Research acheva de tuer les chances de CP/M86 en fixant un prix trop élevé pour son système : $240, soit 4 fois le prix de MS-Dos… Le prix fut ensuite réduit mais il était trop tard, le train était passé. Digital Research essaya de revenir sur le marché du logiciel pour PC à l'occasion de la "course à l'interface graphique" où il prit un bon départ avec GEM (la première version de GEM était bien meilleure que Visi On ou Windows 1.0). Mais GEM arrivait trop tôt vis-à-vis de la puissance des PC de l'époque et cet avantage du "premier arrivé" ne lui servit de fait pas à grand chose : le timing est bien plus important dans la conquête d'un marché car il faut d'abord que ce dernier soit mûr… GEM fit tout de même un parcours intéressant sur les machines Atari 520 et 1024ST.

Dans un premier temps, Microsoft ne réalisa pas tout de suite que MS-Dos représentait son avenir et sa mine d'or. L'éditeur de Redmond, dès 1980, avait prix une licence d'Unix auprès d'AT&T et avait adapté une version de l'OS pour le PC : Xenix. Dans la version 2.0 de MS-Dos (1983), Microsoft mettait même en avant les voies de migration existantes entre Dos et Xenix, pensant toujours que ce dernier était mieux taillé pour les PC plus puissants qui s'annonçait à l'horizon…
Finalement, c'est le tournant vers Windows (un développement difficile et qui dura des années) qui convaincra Bill Gates et son équipe de se concentrer sur le Dos… Si leur stratégie à l'époque semblait vague, elle était très bien pensée: miser sur tous les chevaux. Microsoft était derrière MS-DOS, développait OS/2 pour IBM, s'est procuré un Unix sur PC, et développait Windows pour son propre compte. Bill Gates a même tenter de passer un accord avec Apple pour que ce dernier licencie MacOS, mais Steve Jobs a refusé.

Vu d'aujourd'hui, la domination de Microsoft est si écrasante qu'il peut sembler qu'il en a toujours été ainsi alors qu'en fait, cette conclusion est arrivée seulement avec l'avénement de Windows et que toute l'ère de MS-Dos a été terriblement compétitive. Ainsi, on vit les différents leaders des catégories monter au sommet et dégringoler tout aussi vite suite à leurs propres erreurs.
On a déjà évoqué la trajectoire de Wordperfect dans la précédente section mais nous pouvons y revenir pour illustrer ce renouvellement rapide sur ce marché. Le problème N°1 de Wordperfect, c'est qu'il n'était pas en contrôle de son destin. Son produit était

acheté et non vendu… Donc, quand le marché s'est retourné, la société s'est retrouvé fort dépourvue !
WordPerfect était le roi sous Dos et a su le rester pendant des années mais quand Windows 3.0 a changé le contexte du marché (à l'avantage de Microsoft mais cela ne s'est révélé comme tel que progressivement), les clients se sont mis massivement à réclamer des versions adaptées à cet environnement graphique et l'éditeur de l'Utha n'avait rien à proposer !
La conversion vers le mode GUI (Graphical User Interface) imposé par Windows était difficile car ils l'ont faite trop tard et trop en panique (ce qui impliqua des délais non-réalistes et donc non-tenus, déceptions, impatience du marché et ainsi de suite).

WordPerfect était en contrôle de son destin quand elle se contentait de vendre son produit sur les minis de DG parce que là, le produit était effectivement vendu. La croissance était calme et le volume des ventes n'était pas formidable mais elle en vivait (mais elle était fragile car ce marché était étroit). La bascule sur le marché PC a été brutale et addictive car les ventes étaient plus faciles et les volumes bien plus grands.

Quand il a s'agit de choisir un camp face à l'affontrement Windows-OS/2 qui s'annonçait, WordPerfect a fait l'erreur de croire dans les assurances d'IBM vis-à-vis d'OS/2 (mais ils sont nombreux à avoir la même erreur). On peut comprendre que WordPerfect, Lotus et les autres leaders du marché des applications pur PC aient préféré rallier la bannière d'IBM car ce dernier n'était pas compétitif sur le marché des applicatifs (et donc, ne risquait pas de devenir un compétiteur sérieux… IBM n'avait pas du tout compris le développement sur PC et raisonnait encore en terme de volume de lignes de code pour évaluer un projet… plus il était gros et mieux c'était ! alors que le contraire était plus logique) alors que Microsoft avait déjà prouvé être un compétiteur agressif sur le marché des applicatifs (même si seuls Basic et Excel avaient connu le succès sur leurs marchés respectifs).
Donc, WordPerfect et Lotus préféraient prendre le risque de devenir dépendants d'IBM que de donner encore plus de pouvoir à Microsoft... Ce raisonnement se tenait mais l'erreur a été de croire qu'IBM pouvait mener à terme dans des délais raisonnables un développement tel qu'OS/2. Car, le critère essentiel ici était le "time to market" (le délai de mise sur le marché). Une fois que Windows était effectivement disponible, la "fenêtre d'opportunité" commençait à se refermer et il restait peu de temps pour le concurrencer. Il ne s'agissait plus d'un projet qu'on pouvait mener à sa guise pour un marché captif comme celui des mainframes (où les clients pouvaient bien attendre des années avant qu'une version de DB2 soit utilisable) mais d'un secteur très dynamique à la croissance explosive. Dans ce cas, les clients n'attendent personne, même pas IBM ni les leaders habituels.
Soit votre produit de référence était "prêt pour Windows", soit vos clients passaient à autre chose... Aussi simple que cela !

==========
Encadré sur l'évolution de la bureautique par Louis Naugès, l'inventeur même du mot !

Petite histoire de la Bureautique.
Bureautique 2.0 : de la bureautique individuelle à la "Participatique".
Louis Naugès, Président, Revevol
www.revevol.eu

Introduction
Un milliard de personnes travaillent quotidiennement avec des outils bureautiques qui sont nés avec les PC, il y a 25 ans, au milieu des années 80 : Office et Exchange de Microsoft, Lotus Notes d'IBM... Au cours de la décennie 2010 - 2020, tous vont migrer vers des outils bureautiques de nouvelle

génération, nativement collaboratifs, qui viennent du monde du Web, auxquels j'ai donné le nom de "Participatique". Une petite histoire de la bureautique ... Fin des années 1970 : pour traduire "Office Automation", je propose le mot "Bureautique"; il ... entre dans les dictionnaires moins de dix ans après. Au delà du mot, le concept c'est imposé avec une vitesse impressionnante.

Je vais l'illustrer avec les outils d'écriture, mais j'aurais aussi pu utiliser les outils de calcul. Sur le schéma 1, j'ai représenté quelques générations récentes de ces outils, sans remonter à l'époque du papyrus ! (Insérer image 1, histoire bureautique) - 1910 : les machines à écrire mécaniques commencent à se démocratiser et les claviers QWERTY ou AZERTY s'imposent. - 1960 : IBM "révolutionne" la machine à écrire avec la "balle de golf" qui permet, pour la première fois, de changer de polices de caractères. - 1975 : Wang, Vydec et de très nombreux autres entreprises, depuis disparues, inventent les premières machines de traitement de texte, ordinateurs dédiés pour cette fonction.
- 1978 : de grands acteurs de l'informatique comme IBM ou DEC (Digital Equipment), avec la solution "All-in-One", proposent des logiciels bureautiques installées sur des ordinateurs centraux. Elles seront très vite marginalisées par les progrès des logiciels sur PC. - 1980 : Wordstar, puis Wordperfect deviennent les logiciels de référence pour le traitement de texte : c'est la première fois qu'un logiciel s'impose à un matériel dans les activités d'écriture. - 1983 : Visicalc et Lotus 1,2,3 ; le tableur va révolutionner la vie des financiers et assurer le succès des premiers ordinateurs personnels, et en particulier celui des PC. - 1993 : Microsoft amorce sa domination de la bureautique du poste de travail avec la suite Office, qui crée l'offre intégrée en regroupant tableur, traitement de texte et présentation. Il n'est pas inutile de rappeler que les premières versions d'Office étaient disponibles pour le Macintosh d'Apple ! En créant une version initiale, la première en mode graphique, pour le Macintosh, Microsoft peut devancer ses concurrents quand il lance Windows et assurer ainsi définitivement sa domination sur les outils bureautiques pour PC. L'apogée de la "Bureautique 1.0", fin 2009 750 millions de personnes dans le monde utilisent quotidiennement Office de Microsoft pour écrire, préparer des tableaux et des présentations ; si l'on y ajoute tous les utilisateurs professionnels de versions "non officielles", en clair piratées, on dépasse largement le milliard de personnes. Grâce à cet extraordinaire succès, Microsoft engrange environ 12 milliards de dollars de bénéfices chaque année. C'est la première fois qu'un logiciel atteint une telle masse critique d'utilisateurs ; dans leurs activités professionnelles, c'est très souvent l'outil avec lequel ils passent le plus de temps.

Un deuxième outil bureautique a eu encore plus de succès, la messagerie électronique. Le premier message courriel fut envoyé en 1971 ; fin 2009, Internet réunit 1, 8 milliard d'internautes, donc 1,8 milliard d'utilisateurs de messagerie. Dans les entreprises, deux logiciels de bureautique partagée (messagerie, agenda...) se sont imposés : - Lotus Notes, créé en 1990 par Ray Ozzie. Depuis, Lotus a été racheté par IBM en 1995 et Ray Ozzie est devenu le CSO (Chief Software Architect) de Microsoft. - Exchange de Microsoft, lancé en 1996. Dans les grandes entreprises, ces deux outils ont des parts de marché proches. La véritable révolution de la bureautique de ces 20 dernières années aura été la banalisation de l'usage des outils. A l'époque des machines à écrire, logiciels de traitement de texte, et autres Telex, seuls des spécialistes, en l'occurrence les dactylos et secrétaires, avaient la maîtrise des outils et les cadres continuaient à utiliser le crayon et le papier. Je me souviens de mes combats pour imposer, sans succès, l'apprentissage de la maîtrise du clavier dans les écoles et les universités, comme cela se pratique en Suisse ou aux USA. La France est, en 2009, l'un des rares pays développés où l'apprentissage

du clavier n'est pas obligatoire dans les écoles. Quand on pense aux millions d'heures perdues chaque année par 20 millions de personnes qui pianotent à 2 doigts alors qu'il suffit d'une vingtaine d'heures d'apprentissage pour être capable d'écrire à 10 doigts sans regarder son clavier, on se dit qu'il n'est pas nécessaire de rechercher très loin des gisements de productivité dans nos organisations ! Aujourd'hui, à part quelques "analogistes" attardés, dans des fonctions de Direction de grandes organisations bureaucratiques, tous les cadres et dirigeants utilisent en direct messagerie, tableur ou traitement de texte. 2010 : la "Bureautique 2.0" amorce sa domination Comment, simplement, définir la Bureautique 2.0 ? En allant à l'essentiel, il existe trois caractéristiques "nécessaires" pour qu'un outil logiciel puisse être considéré Bureautique 2.0 : - Accès depuis un navigateur.

Cette condition est indispensable pour deux raisons :
• Indépendance technique de l'objet d'accès. Je dois pouvoir accéder à mes documents, mes messages depuis un PC, un Macintosh, un smartphone, un netbook...
• Permettre l'accès à mes outils depuis mon bureau, mon domicile, un PC en libre service dans la salle d'attente d'un aéroport, un hôtel, et ceci 24h/24, 7j/7.

Les navigateurs modernes, Firefox, Chrome, Safari, Opera ou IE8 répondent bien aux attentes des utilisateurs dans ce domaine. De plus en plus, le respect des standards du Web et les performances techniques, en particulier pour le code JavaScript qui s'exécute dans le navigateur, seront les critères de choix d'un navigateur. - Mode collaboratif natif. Dans la culture du Web 2.0, tous les collaborateurs d'une organisation doivent avoir la possibilité de créer du contenu, d'exprimer leurs opinions. Lorsque je crée un document, un présentation, dans l'immense majorité des cas, je vais le faire avec d'autres personnes, internes et externes à mon entreprise, qui participent avec moi à ce projet comme client, fournisseur ou partenaire. Il est par exemple de plus en plus fréquent, pour Revevol, de partager une proposition commerciale avec nos clients ; ils peuvent, directement proposer des modifications, y compris financières, aux propositions que nous lui envoyons. Quand tout est OK, il suffit de transformer ce document en PDF pour en avoir une version "officielle". - Proposer des fonctions universelles. Les outils bureautiques sont, par nature, universels ; ils ne dépendent ni du secteur d'activité de l'entreprise, ni du métier de la personne, ni de la taille ni du pays où est installée l'entreprise. Les fonctions historiques de la bureautique sont bien connues : traitement de texte, messagerie... On verra plus loin que de nouveaux usages bureautiques innovants apparaissent dans les outils Bureautique 2.0. L'offre de solutions Bureautique 2.0 A l'aube de la décennie 2010, les solutions de Bureautique 2.0 ont atteint un niveau de maturité suffisant pour que toute entreprise, quelle que soit sa taille, puisse trouver une solution qui s'adapte bien à ses attentes. Ce sont des offres encore jeunes, en perpétuelle évolution et qui s'améliorent toutes les semaines. Le choix d'un fournisseur se fera au moins autant sur sa capacité à évoluer et à enrichir ses offres que sur les fonctionnalités précises du produit à un instant donné.

Pour simplifier vos choix, je vous propose de classer les produits en deux familles :
• Solutions Intranet, que l'on installe dans son infrastructures, sur ses serveurs. C'est souvent le premier réflexe d'une entreprise : rechercher une solution qui remplace l'existant et s'installe à l'abri du firewall. Zimbra (http://www.zimbra.com), solution Open Source rachetée depuis par Yahoo!, puis revendue en 2010 à VMWare, est le leader dans ce domaine. Je

pronostique que ces solutions Intranet vont rapidement se marginaliser et disparaître du marché.
• Solutions SaaS, Software as a Service, hébergées sur le "Cloud".

Ces solutions sont commercialisées comme un service, l'entreprise n'ayant rien à installer dans son infrastructure. Ces solutions vont rapidement s'imposer, une fois que les entreprises auront compris que la sécurité de ces produits est supérieure à celle que propose les solutions en interne. Ce sont aussi les seules solutions qui permettent une véritable collaboration entre une entreprise et ses clients, partenaires et fournisseurs. Le futur leader mondial de la Bureautique 2.0 se nomme ... Google. Google Apps Premier Edition (http://www.google.com/apps/intl/fr/business/index.html) est une solution commercialisée depuis le début 2007. Elle est aujourd'hui déployée dans des centaines d'entreprises en France, depuis des PME de 10 personnes jusqu'à de grandes entreprises avec plus de 30 000 utilisateurs. Revevol est depuis 2007 partenaire de Google pour Google Apps. IBM, Cisco, Microsoft commencent à proposer des solutions proches de celles de Google et vont maintenir un fort niveau de concurrence sur ce marché. C'est une excellente nouvelle pour les entreprises car elles vont enfin échapper à la situation de monopole qu'elles ont subie depuis près de 20 ans. Migrer vers la Bureautique 2.0, aujourd'hui ! Rarement, un choix technique aura été plus facile pour les responsables informatiques ! Toutes les entreprises peuvent, doivent commencer à remplacer leur solutions "historiques" par des solutions 2.0. Sur le schéma 2, j'ai représenté l'historique des solutions de messagerie : (Insérer schéma 2 : évolution diligence voiture) - En 1990, Lotus Notes a révolutionné le marché mondial en proposant une solution très innovante, collaborative et multiplaformes. - En 1996, Microsoft a contre-attaqué avec la solution Exchange, qui ne fonctionnait que sur des infrastructures Microsoft, telles que Windows Server ou l'annuaire Active Directory. C'était d'excellentes solutions, tant que l'informatique était dominée par les architectures Client/Serveur. Avec l'explosion du Web, il a fallu attendre la fin des années 2000 pour que les premières solutions Bureautique 2.0 apparaissent. - En 2007, à Paris, Google a annoncé le lancement mondial de la première version de Google Apps pour les entreprises. Comme le montre ce schéma, c'était une version très "rustique", à l'image des premières Citroën 2CV, mais elle démodait définitivement toutes les autres solutions du marché. Aujourd'hui, fin 2009, cette offre a fait des progrès spectaculaires et ressemble plus à la nouvelle Citroën DS3 présentée au salon de Francfort. Qui, en 1960, aurait acheté une diligence pour assurer le transport du courrier ? Qui, en 2013, achètera une solution de Bureautique 1.0 ? C'est la même question ! Bureautique 2.0 : au delà des changements technologiques, de profondes mutations dans les usages. La migration technique est la première étape, la plus facile, la plus évidente et la plus rentable du processus de passage à la Bureautique 2.0. Reste le plus dur, mais le plus passionnant : préparer tous vos collaborateurs à de nouveaux modes de travail, à utiliser efficacement les nouveaux outils collaboratifs mis à leur disposition. Sur le schéma 3, j'ai représenté les trois principales mutations dans les usages : - Du "Je" au "Nous". Cette dimension collaborative, participative, va s'imposer dans tous les usages de la Bureautique 2.0. Tout document, tout tableur, sera nativement créé pour être partagé. La fin des "pièces jointes", ce cauchemar actuel, est proche ! - Du "Je classe" au "Je cherche". Un moteur de recherche est le deuxième outil le plus utilisé sur le Web, après la messagerie. Tous les outils de Bureautique 2.0 proposent des moteurs de recherche performants qui permettent de ne plus classer messages, documents ou présentations. L'utilisateur de ces outils peut toujours, quand il le juge nécessaire, utiliser une ou plusieurs étiquettes (Tags) pour l'aider à retrouver ses documents. La fin des "dossiers", dur à absorber pour de nombreuses personnes qui ont toujours

travaillé de cette manière ! - La généralisation des liens, l'"Hyper-information". Une fois encore, le Web sert de modèle aux usages Bureautique 2.0. Au lieu d'insérer un graphique, une image, une référence, tous les documents Bureautique 2.0 utiliseront des liens "hyper-textes. L'un des principaux avantages de cette troisième mutation est que l'information restera en permanence pertinente. Si je change une donnée dans un tableau, tous les autres documents qui y font référence auront accès à des informations à jour.

Au delà de la Bureautique 2.0, l'émergence de la "Participatique".
En 2009, toute la presse américaine a commencé à utiliser l'expression "Social Computing" pour identifier les nouveaux usages collaboratifs dans les entreprises. Dans mon blog, j'ai proposé de traduire cette expression par "Participatique". (http://nauges.typepad.com/my_weblog/2009/03/participatique-.html) Au delà du mot, l'essentiel est bien entendu de comprendre ce que recouvre ce concept, et les impacts qu'il aura sur les entreprises. Les outils de Participatique ont tous des caractéristiques communes :
• Ils sont universels : tout pays, tout secteur d'activité, toute fonction dans l'entreprise, toute taille d'entreprise.
• Ils permettent de "participer", de contribuer, de créer du contenu.
• Ils sont en mode Web, accessibles par un navigateur.
• Ils sont disponibles sur le "Cloud", en mode SaaS, Software as a Service.

Les nouveaux outils "Participatique", en complément de la Bureautique 2.0, sont :
• Les blogs et microblogs (Tweeter).
• Les Wikis.
• Les Pages d'accueil personnalisées : Netvibes, iGoogle...
• Les réseaux sociaux : LinkedIn, Viadeo, Facebook ...
A l'exception de la Bureautique 2.0, la majorité de ces outils correspondent à de nouveaux usages, dans le monde professionnel. Ils sont déjà très répandus dans le grand public et les "digital natives", de la génération Y, qui arrivent aujourd'hui dans nos bureaux ne comprennent pas pourquoi des entreprises, des DSI, des RSSI (Responsable de la Sécurité des SI) en interdisent l'usage.
=========

Et comme OS/2 version graphique tardait à être disponible, tous les grands acteurs qui avaient fait confiance à IBM se retrouvaient obligés de changer leurs plans à la dernière minute... Et de mettre en veilleuse leurs projets de portage de leurs produits phares vers OS/2 pour se rabattre sur un développement vers Windows que la plupart n'avaient même pas encore entamé !
Tout cela pris du temps, les permières versions de ces projets étaient évidemment décevantes et, pendant ce délai bienvenu, Microsoft engrangeait les ventes et accumulait les parts de marché. Comment se fait-il que Microsoft ait été prêt à temps et su profiter ainsi pleinement de "l'effet Windows" pour assoir la domination que l'on connait aujourd'hui ?

Microsoft fit ses devoirs et appris ses leçons grâce au Mac !
Tout d'abord, il faut rappeler que Microsoft fit de nombreuse tentatives sous Dos pour imposer ses produits dans les principaux segments. Mais MS Word ne pu jamais vraiment menacer WordPerfect dans le segment des traitements de texte sous Dos et MS Multiplan restait un produit marginal face à Lotus 123 dans celui des tableurs. Alors, pourquoi cette situation déprimante a-t-elle changé du tout au tout lors du passage au mode graphique ?

Tout simplement parce que Microsoft consacra des années à s'adapter à la culture de l'interface graphique en commençant à commercialiser ses produits pour le Macintosh d'Apple ce que les autres négligèrent de faire ou ne firent que trop peu d'efforts pour cette plate-forme dont les parts de marché était effectivement très réduites face aux PC sous MS-Dos...
Microsoft n'eut pas cette réticence car Bill Gates avait compris en même temps que Steve Jobs que l'interface graphique représentait l'avenir.
Dans un premier temps, Microsoft fit les mêmes erreurs que les autres : la firme de Redmond se contenta de porter son traitement de texte Word pratiquement tel quel sur le Macintosh en 1984 et ce fut un échec. La version 3.0 de Word pour le Mac connut le succès justement parce qu'il s'agissait de la première mouture qui respectait fidèlement l'interface graphique du Macintosh... La leçon fut comprise et retenue au sein des équipes des développeurs de Microsoft. Cela pu se voir avec Excel qui sortit d'abord pour le Mac (en 1985) avant d'être disponible pour Windows en 1988 (il s'agissait alors de la version 2 de Windows puisque la version 3 n'apparut qu'en 1990).
L'impact d'Excel ne fut pas négligeable car c'était la seule application qui justifiait d'essayer Windows 2.1 sur PC... Le programme était effectivement très réussi et même sous Windows, Excel avait des années d'avance sur Lotus 123 en terme de facilité d'usage.

==========
La culture Microsoft

Si Microsoft a également été co-fondé par Paul Allen, c'est Bill Gates qui a plus influencé Microsoft - en partie parce qu'Allen a du se retirer en 1983 du fait de problèmes de santé. Bill Gates est de notoriété publique hyper-compétitif. Quel que soit le jeu, aussi insignifiant soit-il, il veut gagner. Bill ne veut pas être le meilleur mais le numéro un. Aux débuts de Microsoft, il y avait un pari au sein de la compagnie: qui partirait le dernier pour l'aéroport sans louper son vol ? Bill a gagné en garant sa voiture devant le terminal et non pas au parking. Cet épisode est caractéristique du géant du logiciel : gagner à tout prix, même si le coût ne justifie pas la victoire (comme sa voiture embarquée à la fourrière). Et Microsoft s'est rendu célèbre pour des coups fourrés plus ou moins légaux. A tel point que ces pratiques lui a attiré plusieurs procès pour abus de position monopolistique. Pendant leur procès contre la justice américaine, Microsoft a même été jusqu'à truquer une vidéo qu'ils ont produit pour tenter de prouver un de leurs arguments.

Une autre conséquence de cette hyper-compétitivité est que Microsoft *doit* avoir un adversaire. C'est presque vital pour Redmond. Repérer un nouveau marché, combattre les géants établis sur ce marché et les renverser est presque sa raison d'être. Cela se traduit pour la compagnie par un mélange d'excitation et de peur. Lors de la guerre entre Microsoft et Netscape pour le contrôle du navigateur Web, les développeurs d'Internet Explorer travaillaient jour et nuit - ils avaient tous leur sac de couchage dans leur bureau. La lutte contre Netscape n'était pas simplement gagner un nouveau marché mais était vécue comme une question de survie pour Redmond. Une war room a même été mise en place. Les slogans de Marc Andressen (surtout ceux visant Microsoft) étaient placardés sur les murs. Cette obsession paranoïaque est peut-être exagérée, mais c'est ce qui fait avancer Redmond. On disant pendant un temps que Bill Gates connaissant ses adversaires mieux qu'ils ne se connaissaient eux-même. Ce n'est peut-être plus vrai (surtout depuis que Bill s'est retiré de la direction active de l'éditeur), mais Microsoft sous-estime très rarement ses adversaires.

Bill Gates est à la fois un geek et un businessman. Il peut discuter des avantages et inconvénients d'IPv6 et la seconde d'après parler stratégie de haut

niveau. Son point fort est cependant son côté businessman. Autant Steve Wozniak, co-fondateur d'Apple, était connu pour être un génie de l'électronique, autant Bill Gates n'a jamais été connu pour être un génie de la programmation. Il était sans doute d'un bon niveau, mais pas du niveau d'un Sergey Brin ou d'un Larry Page qui ont conçu un moteur de recherche qui dépassait tout ce qui existait. La véritable force de Gates c'est son sens des affaires. Sur ce point, c'est un excellent businessman. L'épisode de la création du PC est révélateur. Lorsqu'en 1980 IBM a cherché un système d'exploitation pour son futur PC, ils ont tout d'abord contacté Microsoft. N'ayant que le langage (MS-BASIC) mais pas de système d'exploitation, Bill Gates les a invités à contacter Gary Killdal, fondateur de Digital Research, qui avait un système d'exploitation: CP/M. Mais lorsque IBM est venu frapper à la porte de Killdal, ce dernier les a envoyés paître! Microsoft a donc proposé à IBM de s'occuper du système d'exploitation. Paul Allen, co-fondateur de Microsoft, connaissait une petite société locale, Seattle Computer Products, qui avait développé un système d'exploitation pour Intel nommé QDOS (pour Quick & Dirty Operating System). Pour $25000 Microsoft a acquis les droits de vendre QDOS à un seul fabriquant d'ordinateur - mais s'est gardé de dire qu'il s'agissait d'IBM. Il a acquis par la suite les droits complets pour $50000 de plus.

On voit la différence entre Seattle Computer et Microsoft. D'un côté une société qui a développé un produit qu'en bons geeks ils ont appelé "*Quick & Dirty*" ("vite fait, mal fait") et qui l'ont vendu pour $75000, sans doute pensant faire une bonne affaire. De l'autre côté une société qui a renommé le même produit PC-DOS (Personal Computer Disk Operating System) et bâti un empire avec. On voit également la différence entre Digital Research qui n'était pas plus intéressé que ça de faire affaire avec IBM (Digital Research a par exemple refusé de signer l'accord de confidentialité que demandait Big Blue) et Microsoft qui était prêt à tout pour traiter avec IBM, voyant le marché que ce dernier pouvait leur apporter. Microsoft a signé tous les accords de confidentialité que Big Blue demandait et s'est décarcassé pour leur trouver un système d'exploitation.

Il est intéressant de noter que s'il est un homme d'affaire hors pair, Bill Gates n'est *pas* un visionnaire, contrairement à ce que beaucoup de gens pensent. Microsoft s'est lancé dans le marché des systèmes d'exploitation non pas parce qu'ils ont vu un gros marché mais uniquement pour faire plaisir à IBM, ayant peur que ce dernier trouve une autre compagnie à qui acheter un BASIC. Plus tard, dans son livre The Road Ahead (1995), Bill a complètement ignoré l'avènement d'Internet. Tout comme il a été pris de court par la montée en puissance de Google. La seule exception à la règle est le tout premier marché visé par Microsoft, lorsque Gates et Allen ont décidé d'écrire des logiciels dés que le tout premier micro-ordinateur est sorti (l'Altair en 1975).

De même, Bill Gates (comme Steve Ballmer) n'a en effet pas *du tout* le sens du grand public. Bill est beaucoup plus à l'aise sur des marchés de professionnels. C'est ainsi qu'il a loupé la montée du Web, alors grandement grand public (si le Web avait eu dés le début une application professionelle, il est à parier qu'il aurait été beaucoup plus intéressé). Il a raillé pendant des années les balladeurs MP3 et l'iPod pendant que ce dernier devenait un produit culte - Microsoft a finit plus tard par lancer son propre balladeur MP3. Et Microsoft a mis des années avant de voir l'aspect grand public des smartphones. Cela ne veut pas dire que Microsoft est incapable de s'attaquer à des marchés grand public. Il a par exemple réussi à se faire une place au soleil sur le marché des consoles de jeux vidéos (avec la Xbox). Cela veut dire que Redmond est beaucoup plus réactif sur les marchés professionnels que les marchés grand public. Toute vision concernant un marché grand public aura plus de mal à faire

son chemin jusqu'à la direction, et aura nettement moins de chance d'être vue comme stratégique. C'est sans doute pour cette raison que la plupart des jeux vidéos vendus par Microsoft (Flight Simulator, Halo) n'ont pas été développés en interne mais ont été conçus par des sociétés tierces rachetées par Redmond.

En parallèle avec son sens aigu des affaires, Bill (et par extension Microsoft) est fortement motivé par l'argent. Les porte-parole de Gates affirment que la seule chose qui motive ce dernier est de concevoir des bons produits. Il n'empêche : l'un de ses jeux favoris à l'école était le poker (on retrouve une fois de plus son aspect compétitif). Bill n'a pas vu initialement l'intérêt d'Internet, considérant ce dernier comme un réseau où tout était gratuit. "Il n'y a pas d'argent à se faire. En quoi est-ce un business intéressant?" a-t-il répondu à un de ses employés qui tentait de le convaincre de l'utilité du Réseau des réseaux. Microsoft ne s'est intéressé à Google que lorsque celui-ci a commencé à dégager d'importants bénéfices. Et l'action de Microsoft est visible partout au sein de la compagnie. Ceci est à contraster avec Google où ils infligent une amende à tout employé pris en train de jeter un coup d'œil à l'action Google.

Bill Gates n'est pas un visionnaire comme Steve Jobs. Il ne sait pas voir un marché avant tout le monde. Microsoft a donc dû développer d'autres compétences pour compenser. Parfois Redmond a eu de la chance. Cela a été le cas lorsque IBM a frappé à sa porte pour construire son PC. Mais la chance seule ne suffit pas au long terme. Microsoft a donc du apprendre à conquérir des marchés déjà occupés plutôt que d'être un pionnier - chose facilité par l'aspect hyper-compétitif de la compagnie. Et Redmond a un bon palmarès en la matière. Les méthodes employées ont été multiples: s'appuyer sur ses produits existants (Windows), continuer d'attaquer jusqu'à ce que les concurrents fassent des erreurs... et parfois en utiliser des méthodes discutables.

Même s'il s'est parfois aventuré sur le marché du matériel avec la Xbox ou le Zune, Microsoft reste fondalement un éditeur de logiciel (d'où son nom), Bill Gates et Paul Allen étant des passionnés de programmation et non d'électronique au moment où la micro-informatique a décollé dans les années 70. Cette caractéristique, combinée à une culture orientée plus professionnelle que grand public, ont fait que Microsoft n'a réellement réalisé son potentiel qu'avec l'IBM PC, ce dernier lui ayant ouvert la porte sur un marché 1) professionnel et 2) suivant un modèle d'intégration horizontale. Avant l'ère MS-DOS, Microsoft s'est correctement développé en vendant MS-BASIC mais n'en n'est pas moins resté dans l'ombre des constructeurs tels qu'Apple ou Commodore.

Mais même en ayant une forte culture logicielle, aucune entreprise de high tech ne peut tout créer en interne et survivre. Sur ce point, Microsoft est agnostique et est une des compagnies qui a utilisé le plus de moyens pour ajouter de nouveaux produits à son catalogue. Redmond a en effet acheté des compagnies pour leur base installée (comme Hotmail en 1997) mais a également racheté de nombreuses petites compagnies pour leur technologie. Il a également démarché des développeur pour concevoir des nouveaux produits. Le cas le plus célèbre est Dave Cutler, architecte du système d'exploitation VMS, embauché pour concevoir Windows NT et par la suite Windows Azure. A noter que Microsoft a été également accusé de nombreuses fois d'avoir initié des fausses tentatives de rachats dans le seul but d'identifier les développeurs-clé des entreprises cibles pour les démarcher après que les négociations de rachat aient "échoué". Fait assez inhabituel, Microsoft a même parfois acheté les droits d'un produit sans pour autant racheter l'entreprise (MS-DOS acheté à Seattle Computer, SQL Server acheté à Sybase). Pour beaucoup d'entreprises, racheter un produit

sans acheter son éditeur implique une absence d'avantage compétitif (vous êtes deux à avoir le même produit). Mais Redmond a su trouver d'autres avantages compétitifs que le produit seul.

Depuis son tout premier produit, Microsoft a du faire fonctionner ses logiciels avec un grand nombre d'architectures matérielles. A l'heure actuelle, pour chaque nouvelle version de Windows, Microsoft doit interagir avec des milliers de vendeurs tiers. Les fabricants de PC ont besoin de Windows pour le pré-installer sur leurs ordinateurs, les fabricants de périphériques ont besoin de mettre à jour leurs drivers (pilotes), les éditeurs de logiciels veulent s'assurer que leurs programmes tournent correctement sur la nouvelle version, et des millions de développeurs sont intéressés de tirer partie des nouvelles fonctionnalités disponibles. Afin de surmonter ce challenge, Microsoft est passé maître dans l'art du partenariat. Un programme OEM qui permet aux constructeurs de PC de préinstaller Windows, le programme Microsoft Developer Network (MSDN), le programme Microsoft Certified System Engineer (MCSE), plusieurs partenariats commerciaux avec les compagnies qui développent des logiciels sur la plate-forme Microsoft, etc. Côté technique, Redmond a dû standardiser l'accès à ses produits en fournissant une interface de programmation publique, afin que des logiciels tiers puissent interagir avec Windows. L'API (Application Programming Interface) Windows permet de savoir comment interagir avec Windows. La couche HAL (Hardware Abstraction Layer) de Windows isole le gros du code du système d'exploitation du matériel, rendant plus facile de supporter de nombreuses configurations matérielles, etc. Au final, Microsoft a appris à gérer des larges projets impliquant des grosses équipes et un très grand nombre de partenaires. Certains détracteurs trouvent que toutes les versions de Windows ont subi des sérieux retards et son plein de problèmes techniques, mais en général Microsoft a fait un très bon travail quand on considère le nombre de partenaires impliqués.

Mais la relation entre Microsoft et ses partenaires est bien plus importante qu'une histoire de compatibilité matérielle ou logicielle. C'est une affaire d'ordre stratégique. Le premier programme de Microsoft a été MS-BASIC, un langage de programmation. A l'époque, le BASIC (de Microsoft ou d'autres) était incorporé dans l'électronique des micro-ordinateurs. Autrement dit, Microsoft était grandement à la merci des constructeurs, car son succès dépendait complètement de leur désir de licencier les produits de Redmond. Microsoft a donc toujours cherché à influencer le plus possible le rapport de force en sa faveur. Bill Gates n'a jamais voulu forger des partenariats d'égal à égal. Microsoft offre souvent des conditions très favorables sur le court terme, mais va toujours chercher à prendre le contrôle de la relation sur le long terme. Cela implique d'une part éviter le plus possible d'être lié à ses partenaires. Bien qu'il ait fait le dos rond à IBM, Microsoft a demandé -et obtenu- de ce dernier qu'il puisse vendre MS-DOS à d'autres constructeurs. D'autre part cela implique essayer le plus possible de lier les constructeurs -faire qu'ils aient besoin de Microsoft. Le contrat OEM de Windows est un exemple typique. Finalement, Redmond veut s'assurer que ses partenaires d'aujourd'hui ne deviennent pas l'ennemi de demain. Microsoft se souvient très bien qu'avoir été un partenaire d'IBM pour construire son PC lui a permit de subtiliser à Big Blue le contrôle de ce dernier. La firme de Bill Gates a donc toujours cherché à éviter que le même sort lui arrive et de garder pour lui la plupart des profits de l'industrie du PC.

Ce but est cependant contradictoire avec des partenariats à grande échelle. En effet, pour travailler avec autant de compagnies, Microsoft a dû standardiser la manière d'utiliser ses produits. Or qui dit standardisation implique un risque de banalisation : rien n'empêche un systéme d'exploitation concurrent de fournir

les mêmes API que Windows pour faire tourner les applications de ce dernier. De fait, les émulations de Windows ont été nombreuses. La solution pour Redmond a été de constamment enrichir les API de ses produits. Plus c'est complexe et plus ça change vite, plus c'est difficile à imiter. La programmation sous Windows n'a jamais été facile et ne le sera sans doute jamais. En rajoutant constamment de nouvelles API, non seulement Microsoft force les développeurs Windows à constamment se mettre à jour mais il a également empêché qu'un quelconque émulateur Windows soit efficace. La seule manière d'utiliser des programmes Windows correctement sous d'autres systèmes d'exploitation est d'utiliser la virtualisation -processus qui utilise une licence Windows.

Mais la culture Microsoft a deux autres traits: d'une part une forte culture d'ingénieur -et donc avec un goût prononcé pour les fonctionnalités. D'autre part, Redmond s'est historiquement fait de l'argent principalement lorsque les utilisateurs achetaient une nouvelle version de ses produits. En d'autres termes, il faut convaincre les utilisateurs que la mise à jour vaut la peine d'être achetée. Pour un développeur cela veut dire rajouter des fonctionnalités. Ces deux facteurs font que Redmond aime ajouter des fonctionnalités à chaque nouvelle version. Or, ajoutez suffisamment de fonctionnalités à un code même bien optimisé et vous obtenez quelque chose de lourd. Tant pis si le résultat est complexe. Maintenant, certains noteront qu'Apple aime rajouter des fonctionalités, et a parfois mis en avant les "centaines de nouvelles fonctionalités" de certaines versions de MacOS X (eux aussi doivent convaincre ses clients de mettre à jour). La différence est qu'à Cupertino les experts en ergonomie ont un grand poids sur les décisions et qu'Apple a une obsession de la simplicité. A Redmond ce sont les développeurs qui ont pignon sur rue. Microsoft ne sait pas faire simple.

Enfin et cela va dans le sens de l'hyper-compétitivité déjà évoqué, Microsoft n'abandonne jamais... C'est un trait de la culture de Redmond qui est souvent oublié mais qui a parfois été essentiel dans certains des succès de Microsoft : l'éditeur avait peu de succès pour ses applications bureautiques sous Dos mais il a su patienter et s'améliorer jusqu'à être prêt à balayer la concurrence lorsqu'il a s'agit de prendre en compte l'environnement graphique (avec l'avénement de Windows 3.0 en 1990). De même, Microsoft a longtemps lutté avec Novell mais son produit Lan Manager ne pouvait pas renverser la domination de Netware... Jusqu'à ce que Windows NT renverse la situation. Et ainsi de suite, on pourrait multiplier les exemples où la patience et la ténacité de Microsoft a fini par emporter la décision. Microsoft n'abandonne jamais la partie sur les marchés qu'elle a décidé de conquérir, il faut en tenir compte au moment d'établir des prévisions sur son avenir !

==========

Office achève d’enfoncer le clou...
A partir de là, Microsoft détenait tous les cartes pour s’emparer du marché quand les conditions seraient favorables... Et cela ne tarda pas. Alors que Lotus et WordPerfect étaient trop occupés à essayer de développer des versions non-graphiques de leurs produits vedettes pour la première version d’OS/2 (qui n’intégrait pas encore l’interface graphique Presentation Manager), Microsoft rôdait déjà sa formule gagnante Office en la proposant pour le Macintosh qui lui versait toujours de plate-forme d’expérimentation. C’est ainsi qu’une première version d’Office pour le Mac était disponible dès 1989 alors que la première version d’Office pour Windows (Office 92) ne fut disponible qu’à partir de septembre 1992.

Lors du lancement de Windows 3.0, Microsoft n'était pourtant pas le seul à avoir compris qu'il fallait proposer tout de suite des versions adaptées fidèlement à son interface graphique puisque Samma avait publié (dès 1988 et pour Windows 2.1) son traitement de texte Ami. Lotus racheta Samma afin de mettre la main sur Ami et ainsi d'avoir un produit valable à proposer sous Windows. Mais cela ne suffit pas à arrêter Microsoft qui pu dérouler son rouleau compresseur bien aidé par les retards de ses principaux concurrents (Lotus, Borland et WordPerfect, tous ont mis beaucoup de temps à proposer des logiciels acceptables pour Windows et les clients n'ont pas attendus, préférant se tourner vers Microsoft avec Word et Excel dans un premier temps puis avec sa suite Office quand celle-ci fut disponible).

Le marché qui était jusque-là très partagé entre les différents acteurs dominants s'est alors résumé à la domination de Microsoft et de sa suite Office, encore accentuée par la sortie simultanée de Windows 95 et d'Office 95. Les anciens adversaires de Microsoft sur le marché de la bureautique ont quasiment disparus ou se sont marginalisés : Lotus racheté par IBM, Borland se repliant sur les outils de développement, WordPerfect racheté par Novell puis par Corel...

En 1995, Microsoft a lancé un plan marketing génial: proposer un accord OEM aux fabricants de PC. Les fabricants qui signaient cet accord devaient s'engager à préinstaller Windows sur *tous* les PC qu'ils vendaient. En contrepartie de quoi Microsoft leur vendait Windows à prix réduits. L'avantage pour les fabricants était double : d'une part obtenir une réduction sur le prix de Windows, et d'autre part pouvoir le préinstaller, évitant aux utilisateurs d'avoir à installer Windows. Mais cet avantage à court terme s'avera être un désavantage à long terme. Lorsque tous les fabricants ont signé un contrat OEM avec Microsoft, ce dernier ne permet plus aucune distinction avec la concurrence. Au contraire, un constructeur se *doit* de signer le contrat - les utilisateurs sont désormais habitué à avoir Windows préinstallé. C'est pour cette raison que Microsoft a pu imposer ses conditions: le logo Microsoft doit être le premier à apparaitre, etc.
Pour Microsoft, cela a permit non seulement de cimenter Windows auprès du public, mais également d'augmenter considérablement ses marges. Redmond n'a en effet qu'à envoyer qu'un seul pack de CD à Dell (coût minime), mais récolte des royalties pour *chaque* PC que Dell vend sans aucun autre coût. C'est en effet à Dell d'installer - et de supporter - Windows sur toutes ses machines.

Microsoft se trouve un nouvel ennemi : Netscape
Alors que l'été 1995 aurait dû être le moment du triomphe final de Microsoft (avec la sortie très attendue de Windows 95), c'est également le moment où un outsider surgit de nulle part pris place sur le marché : avec son introduction en bourse en août 95, Netscape fit une entrée fracassante au sein de la "scène informatique".
L'irruption de Netscape venait de la soudaine explosion de l'Internet et du Web auprès des utilisateurs. Alors qu'il était en croissance discrète depuis bien des années, l'Internet devenait tout d'un coup "the next big thing" grâce à une application qui en simplifiait l'usage : le navigateur Web. Et c'est une start-up minuscule qui c'était emparée de ce segment : Netscape avec son brower "navigator".
Bill Gates s'était aperçu de l'ampleur de la vague tardivement (mais ils sont nombreux dans le métier à s'être fait surprendre...) mais il compris vite que c'était du sérieux. Avant cela, l'opinion générale au sein de Microsoft était qu'Internet était encore trop immature pour le grand public et c'est pour cela que Microsoft préparait un service en ligne (The Microsoft Network ou MSN) façon AOL ou Compuserve destiné à être une alternative à Internet pour les utilisateurs de Windows 95... MSN fut ensuite convertit en service d'accès à Internet.
Le 26 mai 95, il diffusa un mémo au sein de Microsoft intitulé "The Internet Tidal Wave" (Qu'on peut traduire par "l'Internet Raz de Marée") où il expliquait que l'Internet représentait la nouvelle vague et la nouvelle plate-forme d'avenir pour les applications et s'inquiétait de constater que Microsoft n'y était présent nulle part. C'est que durant l'année 95, le cours des choses semblait s'être accélèré : Sun annonçait Java avec le support immédiat de Netscape...

Java était d'abord un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982). Java, c'est la syntaxe du C++ tout en simplifiant cette dernière. C'est ainsi que Java est débarrassé de plusieurs fonctionnalités de C++ qui complexifient la compréhension du langage (héritage multiple, surcharge d'opérateur, etc.).
Après quelques années de travail souterrain, en octobre 1994, Sun dévoila HotJava (un navigateur Web rustique mais capable d'excuter des applets Java) et la plate-forme Java. Java 1.0a fut disponible en téléchargement en 1994 mais la première version publique du navigateur HotJava arriva le 23 mai 1995 à la conférence SunWorld. L'annonce fut effectuée par John Gage, le directeur scientifique de Sun Microsystems. Son annonce fut accompagnée de l'annonce surprise de Marc Andressen, vice président de Netscape, que Netscape allait inclure le support de Java dans ses navigateurs. Le couple Netscape + Java pouvait être vu comme une alternative possible à la plate-forme Microsoft dans un avenir plus ou moins lointain... Mais la presse spécialisée ne s'embarassait pas de précautions techniques et présenta cette nouvelle "plate-forme" comme une arme anti-Microsoft qui justifiait encore plus de s'intéresser à Internet.
A partir de là, l'emballement médiatique était lâché et le Web devint le nouvel enjeu majeur. Microsoft créa Internet Explorer 1.0 grâce à l'acquisition d'une licence de Spyglass Mosaic (à ne pas confondre avec NCSA Mosaic).
Internet Explorer 1.0 fut alors inclus dans le Pack Plus Windows 95 en août 1995. Mais ce n'était encore qu'une réponse limitée en situation d'urgence...
Le 7 décembre 1995 (le choix de la date était intentionel : le 7 décembre, c'est la date anniversaire de Pearl Harbour...), Microsoft présentait sa stratégie Internet à la presse internationale : être présent partout et même prendre une licence de Java à Sun, du jamais vu !

====
Encadré : les début de Java
Source http://assiste.com.free.fr/p/abc/a/java.html

Java est un langage de programmation à part entière dont le projet a commencé en décembre 1990 avec Patrick Naughton, Mike Sheridan, et James Gosling de SUN Microsystems, avec la participation d'IBM, dans le cadre d'un projet nommé "Green". A ce moment là, MS-Dos est le système d'exploitation dominant dans le monde des ordinateurs individuels.

C'est cette petite équipe, la "Green Team", qui est à l'origine de ce qui va changer la face du monde de l'Internet. Oak n'est, au départ, qu'un développement secret à l'intérieur du projet secret "Green Project" de Sun Microsystems qui tente d'imaginer ce que sera la prochaine vague informatique et comment la prendre.

Pour marquer les esprits et insister sur le fait qu'ils travaillaient sur un projet révolutionnaire et secret, il s'enfermèrent dès mars 1991 et durant 18 mois en un lieu anonyme au 2180 Sand Hill Road (dans Menlo Park en Californie), coupant toute communication avec SUN, et en émergèrent le 3 septembre 1992 avec une première démonstration fonctionnelle sur un appareil portable à écran tactile appelé *7 ("StarSeven" - construit par eux sur une base Sparc grâce à un transfuge qui avait rejoint l'équipe de base). Les spécifications du langage sont celles de James Gosling et le langage s'appelle encore Oak.

L'intérêt suscité est réel, en particulier chez les industriels de la télévision câblée qui pensent déjà à *7 et à la vidéo à la demande. Le langage est développé initialement dans cette optique de réseaux de télévisions numériques câblées avec des contraintes énormes en vitesse d'exécution, ressources minimum, fiaibilité absolue sur des systèmes qui doivent être "non stop"

(jamais redémarés)... Un appel d'offre de Times Warner fait sortir l'équipe de sa "clandestinité" et, en novembre 1992, la "Green Team" devient "FisrtPerson" qui s'installe dans de luxueux bureaux du 100 Hamilton Avenue dans Palo Alto. A ce moment là, l'équipe de développement Java se résume à moins de 30 personnes et n'est toujours qu'un service de Sun Microsystems, pas encore une division.

Ce n'est qu'après l'échec des débouchés industriels vers la télévision numérique câblée (Sun perd l'appel d'offres de la Times Warner - en 1992 c'est encore trop tôt - 15 ans plus tard, en 2007 cela commence tout juste à être une réalité balbutiante) que John Gage, James Gosling, Bill Joy, Patrick Naughton, Wayne Rosing et Eric Schmidt se disent, au cours d'un brainstorming de 3 jours, "Pourquoi pas l'Internet" qui commençait à se populariser pour le transfert de documents média, textes, graphiques, audio, vidéo, à travers des périphériques hétérogènes utilisant tous HTML mais sans la capacité d'ajouter des "comportements" à ces documents média (nous sommes en 1993 et Mosaïc vient de sortir - c'est le premier navigateur Internet, le premier interface "agréable" pour l'Internet qui n'existe que depuis 20 ans et ne connaît que le transfert de fichiers par FTP et Telnet). Il est donc fantastique de s'apercevoir que c'est "par un incroyable accident" que Java a envahi le Net !

Ils développent alors un clone de Mosaïc appelé WebRunner (clin d'œil au film Blade Runner) qui s'appellera officiellement, en 1994, HotJava. C'est la toute première fois que des objets s'animent et que du contenu peut être interactif dans un navigateur Internet. Un serveur ne délivre plus "que des données" mais des données et des applications pour les manipuler. C'est une révolution.

Début 1995, une première démonstration publique laisse l'auditoire ébahi mais en mars 1995 il n'y a que 7 ou 8 copies de WebRunner utilisées. L'équipe décide alors de libérer le code source de WebRunner qui est envoyé sur le Net. Les développeurs s'en emparent et c'est l'envolée grâce à la gratuité accélérée par l'annonce du 23 mai 1995. Ce jour là, à la SunWorld 95, John Gage, directeur scientifique de Sun, annonce l'existence de Java en tant que technologie réelle, sortant des laboratoires et Marc Andressen, vice président de Netscape, fait une annonce surprise en déclarant qu'il introduit Java dans son navigateur, Netscape Navigator (navigateur Internet dominant, voir "unique navigateur" à l'époque).

Le 7 décembre 1995 Sun Microsystems annonce que Microsoft prend une licence Java et compte l'implémenter dans Windows 95 et Windows NT (la licence sera signée en mars 1996 pour une durée de 5 ans). Dans une interview au SunWorld Online, Jon Kannegaard, "chief operating officer" de "Sun's Java Products Division", déclare : "La licence Java accordée à Microsoft conduit Java à être un standard de fait...".

Le 9 janvier 1996 la société filiale de Sun, JavaSoft, est crée et en 2 semaines la première version de Java est disponible.
======

La guerre des navigateurs
A partir de début 1996 commença ce qu'on a appelé la "guerre des navigateurs" : l'affrontement entre Netscape et Microsoft par le biais des versions successives de leurs browers respectifs.
Les versions de Netscape Navigator (Netscape Communicator plus tard) et d'Internet Explorer se succédèrent rapidement durant les quelques années qui suivirent. L'ajout précipité de fonctionnalités prit le pas sur le développement de spécifications techniques concertées, et cette guerre eut comme conséquences des navigateurs instables, de faible

respect des standards alors naissants, de plantages fréquents, de trous de sécurité et de beaucoup d'autres problèmes pour les utilisateurs.
Internet Explorer ne fut pas compétitif avant la version 3.0 (1996) qui incluait le support de scripts et les premières implémentations commerciales des feuilles de style en cascade. Internet Explorer 4.0 sortit en octobre 1997. Internet Explorer 4 fut un tournant de la guerre des navigateurs. Il était plus rapide et supportait mieux les spécifications du W3C que Netscape Navigator 4.0.
Mais même si Miscrosoft améliora régulièrement son propre navigateur, ce n'est pas par ses seuls mérites qu'il "gagna cette guerre"... En effet, c'est plutôt Netscape qui la perdit par ses erreurs (comme souvent, les adversaires de Microsoft se sont tirés "une balle dans le pied" au lieu de se concentrer sur leur travail) comme l'ambition de réécrire complétement Communicator en Java. Sur ce point, on peut vraiment constater que Marc Andreessen s'est comporté de manière dogmatique, poussant jusqu'à l'absurde son soutien à Java au moment où ce dernier était encore loin d'être mature (c'est toujours le cas d'ailleurs !). Ce projet de réécriture n'a cessé de glisser dans le temps, monopilisant les maigres ressources de Netscape au moment où il vallait mieux continuer à faire évoluer la version "normale" de Communicator face à la pression croissance de Microsoft... Pour finir par être abandonné devant l'impossibilité d'obtenir des performances décentes. La version "normale" de Communicator, de son côté, n'était pas brillante. Obnibulé par l'idée de transformer son navigateur en une plate-forme de programmation Web et un outil de "groupware", Netscape a transformé Communicator en un outil beaucoup trop lent, plutôt que de vouloir en faire le meilleur navigateur du marché.

Le choix dogmatique, une erreur classique !
Ce type d'erreur dogmatique n'était pas exceptionnelle puisque Philippe Kahn de Borland commit presque exactement la même quelques années avant avec le portage de Paradox vers Windows. Paradox version MS-Dos était une excellente base de données qui avait été mis sur le marché en 1985 par Ansa Software avant d'être racheté par Borland en 1987. Paradox était un excellent logiciel, bien plus moderne et conforme au modèle relationnel que dBase III, le leader du marché alors. Comme les autres acteurs du marché, Borland fut confronté à la question de la bascule vers Windows à la fin des années 80... Et y répondit de la pire manière qui soit : en changeant l'équipe de développement et en imposant l'utilisation de la programmation objet (une exigence ferme de Philippe Khan).
Du coup, le projet Paradox pour Windows ne cessa de glisser et de prendre du retard. C'était d'autant plus dommage qu'à l'époque, il y avait vraiment une place à prendre : Microsoft n'avait pas encore lancé Access (la version 1 ne fut lancé qu'en novembre 1992) et il n'y avait pas de SGBD crédible sous Windows (Microsoft avait un projet nommé "Omega" qui fut annulé au dernier moment... Finalement, le projet Access bénéficia du rachat de FoxPro par Microsoft).

Microsoft fait feu de tous les côtés sur Netscape...
Microsoft a également attaqué sur d'autres fronts : comme d'habitude, l'éditeur N°1 s'est appuyé sur Windows pour installer en standard Internet Explorer, forçant ainsi la main aux utilisateurs. En parallèle, il a fait de nombreuses offres aux divers fournisseurs d'accès Internet en leur proposant beaucoup de logiciels gratuits s'ils fournissaient exclusivement Internet Explorer à leurs abonnés. Redmond a également visé la jugulaire lorsqu'il a livré Internet Explorer et IIS (son serveur Web) gratuitement avec Windows, forçant Netscape de faire de même et sacrifiant ainsi une bonne partie de ses revenus.
On connait la suite: au début, renverser Netscape semblait impossible. Internet Explorer a commencé avec des parts de marché pitoyables. Et puis, petit à petit, IE a grignoté des parts à Netscape, pour finalement devenir le navigateur numéro un.

Netscape disparait du paysage
Ayant perdu la guerre du navigateur, Netscape ne survit pas longtemps à sa défaite et fut racheté par AOL en 1998. Cette fusion était vraiment "contre nature" vu les différences de cultures des deux équipes et, effectivement, il n'en sortit rien de bon.

Microsoft avait réussit à sortir Netscape du marché, Java n'avait pas concrétisé les espoirs placés en lui et il ne restait guère que des adversaires faibles, dispersés ou à l'offre incompléte (comme Novell) face à lui.
C'est alors que le gouvernement américain décida de mener un procès contre l'éditeur pour abus de position dominante, tout comme il avait menacé de le faire à IBM dans les années 70...
Ce procès dura un an pour être finalement quasiment annulé par la nouvelle administration (le gouvernement Clinton avait été remplacé par le gouvernement Bush entre temps...). Et même si la communauté européenne tentait de continuer les poursuites, une fois encore, la firme de Redmond s'en tirait bien !

Entre temps, Microsoft devait faire face à un autre type de concurrence : le mouvement Open Source. Puisqu'aucun éditeur n'était capable de faire trébucher la société fondée par Bill Gates et Paul Allen, la communauté dans son ensemble pouvait sans doute proposer quelque chose de neuf...

Le mouvement Open Source
Depuis 1998, les logiciels à code source ouvert explosent sur le devant de la scène avec des hauts et des bas. Pour le moment, l'irruption du mouvement Open Source dans l'actualité du monde informatique est encore trop souvent considérée comme un phénomène de mode car l'engouement qui l'entoure oscille entre l'enthousiasme et l'incompréhension.

Ce mouvement vient de loin car il y a toujours eu des logiciels qui étaient versés dans le "domaine public" et pour lesquels il n'y avait aucune licence, aucune restriction, même pas d'auteur identifié dans certains cas !
Mais, au-delà de la zone "où tout est permis" du domaine public, un programmeur américain, Richard Stallman, décida de formaliser une démarche rigoureuse où la liberté de copier, de modifier et de distribuer librement des logiciels serait préservée et garantie. Cette réflexion déboucha sur une licence spécifique, la GPL (General Public Licence).

> ===
> Encadré sur la GPL, extraits issus de http://fr.wikipedia.org/wiki/Licence_publique_générale_GNU
>
> La Licence publique générale GNU, ou GNU General Public License (son seul nom officiel en anglais, communément abrégé GNU GPL voire simplement couramment « GPL ») est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU. Richard Stallman et Eben Moglen, deux des grands acteurs de la Free Software Foundation, en furent les premiers rédacteurs. Sa dernière version est la GNU GPL version 3 publiée le 29 juin 2007.
> Elle a depuis été adoptée, en tant que document définissant le mode d'utilisation, donc d'usage et de diffusion, par de nombreux auteurs de logiciels libres.
> La GPL met en œuvre la notion de copyleft, un jeu de mots anglais faisant référence à la notion de copyright (right en anglais signifie à la fois le droit, c'est-à-dire la règle juridique, et la droite, qui marque une direction) que l'on peut transposer en français en parlant de gauche d'auteur par référence au droit d'auteur. Pour autant le copyleft n'est pas l'antithèse du copyright, bien au contraire, puisqu'en réalité le premier s'appuie sur le second. Ainsi le copyleft comme le copyright définissent et encadrent les droits des utilisateurs de façon contraignante. Si le mécanisme est le même, les objectifs différent : là où le copyright garantit exclusivement les droits de l'auteur, le copyleft s'attarde en outre tout particulièrement sur les droits des utilisateurs et vise à préserver la

liberté d'utiliser, d'étudier, de modifier et de diffuser le logiciel et ses versions dérivées.
La GPL est la licence de logiciel libre la plus utilisée.
...
L'objectif de la licence GNU GPL, selon ses créateurs est de garantir à l'utilisateur les droits suivants (appelés libertés) sur un programme informatique :
La liberté d'exécuter le logiciel, pour n'importe quel usage ;
La liberté d'étudier le fonctionnement d'un programme et de l'adapter à ses besoins, ce qui passe par l'accès aux codes sources ;
La liberté de redistribuer des copies ;
La liberté d'améliorer le programme et l'obligation de rendre publiques les modifications afin que l'ensemble de la communauté en bénéficie.

La GNU GPL (abrégé GPL) a été écrite par Richard Stallman pour être utilisée sur les programmes du projet GNU. Elle est basée sur l'assemblage des licences utilisées par GNU Emacs, GNU Debugger (GDB) et la GNU Compiler Collection (GCC). Ces licences contiennent des clauses identiques, mais elles sont spécifiques à chaque programme. Le but de Stallman est de produire une licence unique qui pourra être utilisée pour chaque projet et que cette licence permette au plus grand nombre de projets de partager leur code source. C'est ainsi que naquit la GPL version 1 en février 1983.
En 1990, il était devenu évident qu'une licence moins restrictive serait utile pour quelques bibliothèques logicielles. Ainsi, quand la version 2 de la GPL apparut en juin 1991, une nouvelle licence fut créée, la GNU Library General Public License (abrégé GNU LGPL ou LGPL) prenant elle aussi la version 2 pour marquer leur lien de parenté. Les numéros de versions sont devenus différents en 1999 quand la version 2.1 de LGPL est arrivée. La LGPL a changé de nom en même temps afin de mieux réflèter sa place par rapport à l'esprit GNU : elle s'appelle désormais la GNU Lesser General Public License (toujours abrégé GNU LGPL ou LGPL).
En 2005, Richard Stallman a commencé à écrire la version 3 de la GPL.
...
===

Au début des années 90, Linux commença à apparaitre et certains au sein du mouvement du logiciel libre voulait s'affranchir de la tutelle de Richard Stallman, ce fut le cas notamment d'Eric Raymond qui fut à l'origine du mouvement Open Source qui diffère subtilement du mouvement logiciel libre.
Dans le livre "Richard Stallman et la révolution du logiciel libre", Sam Williams, Richard Stallman & Christophe Masutti raconte ainsi le moment où Eric Raymond rédiga son fameux article "La cathédrale et le bazar" qui fut à l'origine du mouvement Open source :
Raymond s’aperçut que cette méthode permettait en outre de contourner la loi de Brooks. Énoncée pour la première fois par Fred P. Brooks, responsable du projet OS/360 chez IBM et auteur du livre The Mythical Man-Month en 1975, cette loi postulait que l’ajout de développeurs à un projet ne faisait qu’entraîner des retards supplémentaires. Croyant comme la plupart de ses confrères programmeurs que le logiciel, à l’instar de la soupe, a tout avan- tage à être préparé par un nombre limité de cuisiniers, Raymond sentit que quelque chose de révolutionnaire était à l’œuvre. En invitant toujours plus de cuisiniers à la cuisine, Torvalds avait effectivement réussi à rendre le logiciel « meilleur ».

La loi de Brooks est la forme courte de la citation suivante, tirée du livre de l’auteur : « Puisque la fabrication de logiciels est intrinsèquement le fruit de l’effort conjugué de différents systèmes — un exercice d’interactions com- plexes — l’effort de

communication est grand, et il contrebalance rapidement la diminution du temps de travail individuel rendue possible par le partage des tâches. Au lieu de réduire les délais, ajouter plus de personnel les allonge. »

Raymond coucha ses observations sur le papier et les mit en forme dans un discours, qu'il lut rapidement devant un groupe d'amis et de voisins, dans le comté de Chester, en Pennsylvanie. Intitulé La cathédrale et le bazar, le texte établissait un contraste entre le management de type « bazar » initié par Torvalds, et celui de style « cathédrale », utilisé habituellement. D'après Raymond, les réactions furent enthousiastes, et elles le furent encore plus au printemps suivant, lors du Linux Kongress de 1997 qui réunissait des utilisateurs GNU/Linux allemands. Raymond finit par convertir ce discours en article, intitulé lui- aussi La cathédrale et le bazar. Le texte tirait son nom de l'analogie centrale de Raymond. Auparavant, les programmes étaient des « cathédrales », monuments impressionnants dont la construction était planifiée de façon centralisée, et conçus pour résister aux as- sauts du temps. À l'opposé, Linux ressemblait davantage à « un grand bazar bruyant », un projet logiciel développé grâce aux dynamiques floues et décentralisées de l'Internet.

> ===== encadré sur l'article de Raymond =====
> La cathédrale et le bazar (The Cathedral and the Bazaar) 1998
> Auteur : Eric S. Raymond (esr@thyrsus.com) - Traducteur : Sébastien Blondeel
> Extraits de http://www.linux-france.org/article/these/cathedrale-bazar/cathedrale-bazar_monoblock.html
>
> 1. La cathédrale et le bazar
> Linux est subversif. Qui aurait imaginé, il y a seulement cinq ans, qu'un système d'exploitation de classe internationale prendrait forme comme par magie à partir de bidouilles
> Certainement pas moi. Quand, début 1993, Linux est apparu pour la première fois sur mon écran radar, cela faisait déjà dix ans que j'étais impliqué dans le développement sous Unix et dans la programmation de logiciels dont le code source est ouvert. Au milieu des années 1980, j'étais l'un des premiers contributeurs à GNU. J'avais distribué sur le réseau une bonne quantité de logiciels dont le code source est ouvert (nethack, les modes VC et GUD pour Emacs, xlife, et d'autres), encore largement utilisés de nos jours. Je pensais savoir comment tout cela fonctionnait.
> Linux a remis en cause une grande partie de ce que je croyais savoir. J'avais prêché l'évangile selon Unix sur l'utilisation de petits outils, le prototypage rapide et la programmation évolutive, depuis des années. Mais je pensais aussi qu'il existait une certaine complexité critique au delà de laquelle une approche plus centralisée, plus a priori, était nécessaire. Je pensais que les logiciels les plus importants (comme les systèmes d'exploitation et les très gros outils comme Emacs) devaient être conçus comme des cathédrales, soigneusement élaborés par des sorciers isolés ou des petits groupes de mages travaillant à l'écart du monde, sans qu'aucune version bêta ne voie le jour avant que son heure ne soit venue.
> Le style de développement de Linus Torvalds - distribuez vite et souvent, déléguez tout ce que vous pouvez déléguer, soyez ouvert jusqu'à la promiscuité - est venu comme une surprise. À l'opposé de la construction de cathédrales, silencieuse et pleine de vénération, la communauté Linux paraissait plutôt ressembler à un bazar, grouillant de rituels et d'approches différentes (très justement symbolisé par les sites d'archives de Linux, qui acceptaient des contributions de n'importe qui) à partir duquel un système stable et cohérent ne pourrait apparemment émerger que par une succession de miracles.
> Le fait que ce style du bazar semblait fonctionner, et bien fonctionner, fut un choc supplémentaire. Alors que j'apprenais à m'y retrouver, je travaillais dur,

non seulement sur des projets particuliers, mais encore à essayer de comprendre pourquoi le monde Linux, au lieu de se disloquer dans la confusion la plus totale, paraissait au contraire avancer à pas de géant, à une vitesse inimaginable pour les bâtisseurs de cathédrales.
Vers le milieu de 1996, je pensais commencer à comprendre. Le hasard m'a donné un moyen incomparable de mettre ma théorie à l'épreuve, sous la forme d'un projet dont le code source est ouvert et que je pourrais consciemment faire évoluer dans le style du bazar. Ce que je fis -- et je rencontrai un franc succès.
Dans le reste de cet article, je conterai l'histoire de ce projet, et je l'utiliserai pour proposer des aphorismes relatifs au développement efficace de logiciels dont le code source est public. Je n'ai pas appris toutes ces choses dans le monde Linux, mais nous verrons de quelle manière le monde Linux les place au devant de la scène. Si je ne me trompe pas, elles vous aideront à comprendre exactement ce qui fait de la communauté Linux une telle manne de bons logiciels - et à devenir plus productif vous-même.

2. Le courrier doit passer !

Depuis 1993, j'étais aux commandes du service technique d'un modeste fournisseur de services Internet proposant un accès libre, appelé Chester County Interlink (CCIL) et situé dans le West Chester, en Pennsylvanie (je suis l'un des co-fondateurs de CCIL et j'ai écrit notre unique logiciel de ``bulletin-board system'' (BBS, systèmes de discussions entre utilisateurs) multi-utilisateurs -- vous pouvez vérifier cela en accédant par telnet à locke.ccil.org. Aujourd'hui, ce service supporte presque trois mille utilisateurs sur trente lignes.) Ce travail m'a permis d'accéder au réseau 24 heures sur 24, à travers la ligne de 56K de CCIL -- en réalité, c'était presque une nécessité !
De cette manière, je m'étais habitué au courrier électronique instantané par l'Internet. Pour des raisons compliquées, il était difficile de faire fonctionner SLIP entre ma machine située à mon domicile (snark.thyrsus.com) et CCIL. Quand j'ai enfin réussi, j'ai trouvé que me connecter régulièrement à locke pour vérifier que je n'avais pas de courrier électronique était ennuyeux. Je souhaitais que mon courrier me parvienne sur snark de telle sorte que je sois averti dès son arrivée et que je puisse le traiter à l'aide de mes outils locaux.
Indiquer simplement à sendmail de faire suivre le courrier n'aurait pas fonctionné, parce que ma machine personnelle n'est pas sur le réseau en permanence et ne dispose pas d'une adresse IP statique. J'avais besoin d'un programme qui étende une main tentaculaire à travers ma connexion SLIP, et me rapporte mon courrier afin de le distribuer de manière locale. Je savais que de telles choses existaient, et que la plupart d'entre elles utilisait un simple protocole d'application (application protocol) appelé POP (Post Office Protocol, protocole du bureau de poste). Et bien sûr, le système d'exploitation BSD/OS de locke, incluait un serveur POP3.
J'avais besoin d'un client POP3. Alors je suis allé sur le réseau et j'en ai trouvé un. En fait, j'en ai trouvé trois ou quatre. J'ai utilisé pop-perl pendant un moment, mais il lui manquait une fonctionnalité qui semblait évidente, la possibilité de bidouiller les adresses sur le mail rapporté afin que les réponses fonctionnent correctement.
Le problème était le suivant: un dénommé `joe' sur locke m'envoyait du courrier. Si je rapportais le courrier sur snark et tentais d'y répondre, mon programme de courrier essayait en toute bonne foi de le faire parvenir à un `joe' sur snark, qui n'existait pas. Éditer les adresses de retour à la main pour leur ajouter `@ccil.org' est rapidement devenu assez pénible.
C'était typiquement le genre de choses que l'ordinateur devait faire à ma place. Mais aucun des clients POP existants ne savait comment faire ! Et ceci nous

amène à la première leçon: 1. Tout bon logiciel commence par gratter un développeur là où ça le démange.
Ceci est peut-être une évidence (il est bien connu, et depuis longtemps, que "Nécessité est mère d'Invention'') mais trop souvent on voit des développeurs de logiciels passer leurs journées à se morfondre à produire des programmes dont ils n'ont pas besoin et qu'ils n'aiment pas. Ce n'est pas le cas dans le monde Linux -- ce qui peut expliquer pourquoi la plupart des programmes issus de la communauté Linux sont de si bonne facture.
Alors, me suis-je précipité frénétiquement dans le codage d'un client POP3 tout nouveau tout beau, qui soutienne la comparaison avec ceux qui existent déjà ? Jamais de la vie ! J'ai examiné avec beaucoup d'attention les utilitaires POP que j'avais à ma disposition, en me demandant ``lequel de ces programmes est le plus proche de ce que je cherche ?''. Parce que 2. Les bons programmeurs savent quoi écrire. Les grands programmeurs savent quoi réécrire (et réutiliser).
N'ayant pas la prétention de me compter parmi les grands programmeurs, j'essaie seulement de les imiter. Une caractéristique importante des grands programmeurs est la paresse constructive. Ils savent qu'on n'obtient pas 20/20 pour les efforts fournis, mais pour le résultat obtenu, et qu'il est pratiquement toujours plus facile de partir d'une bonne solution partielle que de rien du tout. Linus Torvalds, par exemple, n'a pas essayé d'écrire Linux en partant d'une page blanche. Au contraire, il a commencé par réutiliser le code et les idées de Minix, un minuscule système d'exploitation ressemblant à Unix tournant sur les 386. La totalité du code de Minix a été abandonnée ou réécrite complètement depuis -- mais tant qu'il était là, ce code a servi de tuteur à l'enfant qui deviendrait Linux.
Dans le même esprit, je me suis mis en quête d'un utilitaire POP raisonnablement bien écrit, afin de l'utiliser comme base pour mon développement.
La tradition de partage du code source du monde Unix a toujours favorisé la réutilisation de code (c'est pourquoi le projet GNU a choisi Unix comme système d'exploitation de travail, malgré de sérieuses réserves quant au système d'exploitation lui-même). Le monde Linux a pratiquement emmené cette tradition jusqu'à sa limite technologique; il dispose de téraoctets (millions de millions d'octets) de codes sources ouverts généralement disponibles. De telle sorte que passer du temps à chercher le "presqu'assez bon'' de quelqu'un d'autre a plus de chances de donner de bons résultats dans le monde Linux que nulle part ailleurs.
Et pour moi, cela a marché. Avec ceux que j'avais déjà trouvés, ma deuxième recherche aboutit à un total de neuf candidats -- fetchpop, PopTart, get-mail, gwpop, pimp, pop-perl, popc, popmail et upop. Je me suis d'abord concentré sur le fetchpop de Seung-Hong Oh. J'y ai ajouté ma fonctionnalité de réécriture des entêtes, et j'ai fait un certain nombre d'améliorations que l'auteur a acceptées dans sa version 1.9.
Quelques semaines plus tard, cependant, je suis tombé sur le code de popclient par Carl Harris, et j'ai rencontré un problème. Fetchpop contenait des idées bonnes et originales (comme son mode démon), mais il ne pouvait gérer que POP3 et il était programmé d'une manière quelque peu amateur (Seung-Hong est un programmeur brillant mais inexpérimenté, et ces deux caractéristiques apparaissaient dans son travail). Le code de Carl était meilleur, assez professionnel et solide, mais son programme ne proposait pas certaines fonctionnalités de fetchpop, importantes et plutôt difficiles à coder (en particulier, celles que j'avais programmées moi-même).
Continuer, ou basculer ? Si je basculais, cela revenait à abandonner le travail que j'avais déjà réalisé, en échange d'une meilleure base de développement.

Une raison pragmatique me poussant à changer était la présence d'un support pour plusieurs protocoles. POP3 est le protocole le plus communément utilisé, mais ce n'est pas le seul. Fetchpop et ses semblables n'étaient pas compatibles avec les protocoles POP2, RPOP, ou APOP, et j'envisageais déjà vaguement d'ajouter IMAP (Internet Message Access Protocol, protocole d'accès aux messages par Internet, le protocole le plus récent et le plus puissant), rien que pour le plaisir.
Mais j'avais une raison plus théorique de penser que changer serait une bonne idée malgré tout, une raison que j'avais apprise bien avant Linux.

3. "Prévoyez d'en jeter un, car de toute manière, vous le ferez.'' (Fred Brooks, "The Mythical Man-Month'', chapitre 11)
"Le mythe du mois-homme'', traduction de Frédéric Mora, ITP, ISBN 2-84180-081-4, est un livre où il explique que l'ensemble des croyances relatives à l'unité de mesure "mois-homme'' est un mythe. Ou, dit autrement, on ne comprend souvent vraiment bien un problème qu'après avoir implanté une première solution. La deuxième fois, on en sait parfois assez pour le résoudre correctement. Ainsi, si vous voulez faire du bon travail, soyez prêt à recommencer au moins une fois.
Bien (me suis-je dit), les modifications à fetchpop furent un coup d'essai. J'ai donc basculé.
Après avoir envoyé à Carl Harris mon premier lot de modifications apportées à popclient, le 25 juin 1996, j'ai découvert qu'il avait pratiquement perdu tout intérêt pour popclient depuis quelque temps. Le code était un peu poussiéreux, et de petits bogues y subsistaient. J'avais de nombreuses modifications à apporter, et nous nous sommes rapidement mis d'accord pour que je reprenne le programme.
Avant même que je m'en rende compte, le projet avait monté d'un cran. Je n'envisageais plus d'apporter des modifications mineures à un client POP existant. Je m'engageais à maintenir un client POP dans son intégralité, et je savais que certaines des idées qui germaient dans mon cerveau m'amèneraient probablement à effectuer des modifications majeures.
Dans une culture du logiciel qui encourage le partage du code, il est naturel pour un projet d'évoluer ainsi. J'étais en train d'expérimenter le fait suivant : 4. Si vous avez la bonne attitude, les problèmes intéressants viendront à vous.
Mais l'attitude de Carl Harris était encore plus importante. Il comprit que 5. Quand un programme ne vous intéresse plus, votre dernier devoir à son égard est de le confier à un successeur compétent.
Sans même avoir à en discuter, Carl et moi savions que nous poursuivions un objectif commun: chercher la meilleure solution. La seule question d'importance pour chacun de nous était de nous assurer que le programme serait en de bonnes mains avec moi. Une fois que j'en apportai la preuve, il me fit place nette avec bonne volonté. J'espère agir de même quand mon tour viendra.
3. De l'importance d'avoir des utilisateurs
C'est ainsi que j'héritai de popclient. De même, et c'est tout aussi important, c'est ainsi que j'héritai de l'ensemble des utilisateurs de popclient. Il est merveilleux de disposer d'utilisateurs, et pas seulement parce qu'ils vous rappellent que vous remplissez un besoin et que vous avez fait une bonne chose. Éduqués de façon appropriée, ils peuvent devenir vos co-développeurs. Le fait que beaucoup d'utilisateurs sont également des bidouilleurs est une autre force de la tradition Unix, et c'est une caractéristique que Linux a poussée avec bonheur dans ses derniers retranchements. De plus, la libre disponibilité du code source en fait des bidouilleurs efficaces. Ceci peut se révéler incroyablement utile pour réduire le temps de débogage. Avec un peu d'encouragement (ils n'en réclament pas beaucoup), vos utilisateurs diagnostiqueront les problèmes, suggéreront des corrections, et vous aideront à

améliorer le code bien plus rapidement que vous n'auriez pu le faire, si vous étiez laissé à vous même.
6. Traiter vos utilisateurs en tant que co-développeurs est le chemin le moins semé d'embûches vers une amélioration rapide du code et un débogage efficace.
Il est facile de sous-estimer la puissance de ce phénomène. En fait, la quasi-totalité d'entre nous, appartenant au monde du logiciel dont le code source est ouvert, sous-estimions effroyablement la facilité avec laquelle il s'adapterait à l'augmentation du nombre d'utilisateurs et de la complexité des systèmes, jusqu'à ce que Linus Torvalds ne nous démontre le contraire.
En réalité, je pense que la bidouille la plus ingénieuse de Linus, et celle qui a eu le plus de conséquences, n'a pas été la construction du noyau de Linux en lui-même, mais plutôt son invention du modèle de développement de Linux. Un jour où j'exprimais cette opinion en sa présence, il souria et répéta tranquillement cette pensée qu'il a si souvent exprimée: "Je suis tout simplement une personne très paresseuse, qui aime se faire remercier pour le travail effectué par d'autres.'' Paresseux comme un renard. Ou, comme Robert Heinlein aurait pu le dire, trop paresseux pour pouvoir échouer.
Rétrospectivement, on peut voir dans le développement des bibliothèques Lisp et des archives de code Lisp pour GNU Emacs un précédent aux méthodes et au succès de Linux. Au contraire du coeur d'Emacs, construit en C à la manière des cathédrales, et au contraire de la plupart des autres outils proposés par la FSF (Free Software Foundation, Fondation pour le Logiciel Libre), l'évolution de l'ensemble du code Lisp a été très fluide et très dirigée par les utilisateurs. On a souvent réécrit les idées et les prototypes des modes trois ou quatre fois avant d'atteindre une forme finale stable. Et les exemples de collaborations occasionnelles rendues possibles par l'Internet, à la manière de Linux, ne manquent pas.
En fait, ma bidouille personnelle qui a rencontré le plus de succès, avant fetchmail, fut probablement le mode VC pour Emacs, issu d'une collaboration par courrier électronique à la Linux avec trois autres personnes, et je n'ai toujours rencontré que l'une d'entre elles (Richard Stallman, l'auteur d'Emacs et le fondateur de la FSF) à ce jour. C'était une interface sous Emacs pour SCCS, RCS et plus tard pour CVS, qui fournissait toutes les opérations de contrôle de version avec un seul bouton. Il était parti d'un mode sccs.el très rudimentaire et tout petit, écrit par quelqu'un d'autre. Et le développement de VC fut réussi parce que, à la différence d'Emacs lui-même, le code Lisp pour Emacs pouvait traverser très rapidement les cycles de mise à jour/test/amélioration.

4. Distribuez tôt, mettez à jour souvent
Un élément essentiel du système de développement de Linux est la mise à jour rapide et fréquente des nouvelles versions. La plupart des développeurs (moi y compris) pensait que ce n'était pas une bonne méthode pour des projets de taille non triviale, parce que des versions prématurées sont quasiment, par définition, des versions boguées, et qu'il n'est pas dans votre intérêt d'abuser de la patience des utilisateurs.
C'est cette croyance qui a consolidé l'attachement général au style de développement de type cathédrale. Si l'objectif premier était de présenter aux utilisateurs une version aussi dépouillée de bogues que possible, alors vous ne faisiez une mise à jour que tous les six mois (voire moins souvent encore), et vous travailliez d'arrache-pied au débogage entre les mises à jour. C'est de cette manière qu'a été mis au point le coeur d'Emacs, écrit en C. Ce ne fut pas le cas, en pratique, de la bibliothèque Lisp -- parce qu'il existait des archives Lisp ``vivantes'' hors de portée de la FSF, et où on pouvait trouver de nouvelles versions de code de développement indépendamment du cycle de mises à jour d'Emacs.

La plus importante de ces archives, l'archive elisp de l'État de l'Ohio, était en avance sur son temps et avait déjà deviné l'esprit et la plupart des spécificités des archives Linux actuelles. Mais bien peu d'entre nous réfléchissaient vraiment beaucoup sur ce que nous faisions ou sur les problèmes liés au développement de style cathédrale adoptés par la FSF que l'existence même de cette archive pouvait suggérer. J'ai vraiment tenté, une fois, vers 1992, de faire passer de manière formelle, un bon morceau du code d'Ohio dans la bibliothèque Lisp officielle d'Emacs. Je n'ai rencontré que des guerres d'opinions et je suis rentré bredouille dans les grandes longueurs.

Mais un an plus tard, alors que Linux devenait de plus en plus apparent, il était clair que quelque chose de différent, de plus sain, était en train de se produire. La politique de développement ouvert de Linus était l'antithèse même du modèle des cathédrales. Les archives de sunsite et de tsx-11 bourgeonnaient, de multiples distributions étaient mises à l'eau. Et tout cela était rythmé par une fréquence de mise à jour du coeur même du système jusqu'alors inégalée. Linus traitait ses utilisateurs comme des développeurs de la manière la plus efficace:

7. Distribuez tôt. Mettez à jour souvent. Et soyez à l'écoute de vos clients.

La grande innovation de Linus ne fut pas tant de faire cela (c'était une tradition du monde Unix depuis un bon moment, en tout cas sous une forme proche), que de donner à cette idée un niveau d'intensité correspondant à la complexité de ce qu'il développait. En ces temps reculés (autour de 1991), il lui arrivait de mettre à jour son noyau plusieurs fois par jour !

Comme il cultivait sa base de co-développeurs et recherchait des collaborations sur Internet plus que quiconque jusque là, cela a marché.

Mais comment cela a-t-il marché ? Et, était-ce quelque chose que je pouvais dupliquer, ou cela reposait-il uniquement sur quelque trait de génie propre à Linus Torvalds ?

Je ne le pensais pas. Je vous l'accorde, Linus est un bidouilleur sacrément bon (combien parmi nous pourraient mettre au point l'intégralité du noyau d'un système d'exploitation prêt à être mis en production ?). Mais Linux ne constituait pas vraiment un réel progrès conceptuel. Linus n'est pas (en tout cas, pas encore) un génie de l'innovation dans la conception comme peuvent l'être Richard Stallman ou James Gosling (de NeWS et Java). Au contraire, Linus est à mes yeux un génie de l'ingéniérie, doué d'un sixième sens qui lui permet d'éviter les bogues et les impasses de développement et surtout d'un authentique talent pour trouver le chemin de moindre effort reliant A à B. En effet, la conception de Linux dans son intégralité est imprégnée de cette qualité et reflète l'approche conservatrice et simplificatrice qu'a Linus de la conception. Ainsi, si la rapidité des mises à jour et l'extraction de la substantifique moelle de l'Internet en tant que moyen de communication n'étaient pas des hasards, mais faisaient partie intégrante du côté visionnaire génial de Linus vers le chemin le plus facile à suivre, qu'est-ce donc qu'il maximisait ? Qu'est-ce donc qu'il tirait de toute cette machinerie ?

Posée de cette manière, la question contient sa propre réponse. Linus stimulait et récompensait ses utilisateurs/bidouilleurs en permanence -- il les stimulait par la perspective auto-gratifiante de prendre part à l'action, et il les récompensait par la vue constante (et même quotidienne) des améliorations de leur travail.

Linus cherchait directement à maximiser le nombre de personnes-heures jetées dans la bataille du débogage et du développement, au prix éventuel d'une certaine instabilité dans le code et de l'extinction de sa base d'utilisateurs si un quelconque bogue sérieux se révélait insurmontable. Linus se comportait comme s'il croyait en quelque chose comme:

8. Étant donné un ensemble de bêta-testeurs et de co-développeurs suffisamment grand, chaque problème sera rapidement isolé, et sa solution semblera évidente à quelqu'un.

Ou, moins formellement, "Étant donnés suffisamment d'observateurs, tous les bogues sautent aux yeux.'' C'est ce que j'appelle: "La Loi de Linus''.

Ma première formulation de cette loi était que chaque problème "semblera clair comme de l'eau de roche à quelqu'un''. Linus a objecté qu'il n'était pas nécessaire, et que d'ailleurs ce n'était pas le cas en général, que la personne qui comprenne un problème soit la personne qui l'ait d'abord identifié.

``Quelqu'un trouve le problème,'' dit-il, "et c'est quelqu'un d'autre qui le comprend. Je pousserai le bouchon jusqu'à dire que le plus difficile est de trouver le problème.'' Mais le principal est que ces deux événements se produisent en général vite.

C'est là, je pense, la différence fondamentale sous-jacente aux styles de la cathédrale et du bazar. Dans la programmation du point de vue de la cathédrale, les bogues et les problèmes de développement représentent des phénomènes difficiles, ennuyeux, insidieux, profonds. Il faut à une poignée de passionnés des mois d'observations minutieuses avant de bien vouloir se laisser convaincre que tous les bogues ont été éliminés. D'où les longs intervalles séparant les mises à jour, et l'inévitable déception quand on se rend compte que la mise à jour tant attendue n'est pas parfaite.

Dans le point de vue bazar, d'un autre côté, vous supposez qu'en général, les bogues sont un phénomène de surface -- ou, en tout cas, qu'ils sautent rapidement aux yeux lorsqu'un millier de co-développeurs avides se précipitent sur toute nouvelle mise à jour. C'est pourquoi vous mettez à jour souvent afin de disposer de plus de corrections, et un effet de bord bénéfique est que vous avez moins à perdre si de temps en temps, un gros bogue vous échappe.

Et voilà. C'est tout. Si la "Loi de Linus'' est fausse, alors tout système aussi complexe que le noyau Linux, subissant les bidouilles simultanées d'autant de mains que ce fut le cas du noyau Linux, aurait dû à un certain moment s'effondrer sous le poids des interactions néfastes et imprévues, sous le poids de bogues "profonds'' non découverts. D'un autre côté, si elle est juste, elle suffit à expliquer l'absence relative de bogues dans Linux.

Et peut-être bien, après tout, que cela ne devrait pas être aussi surprenant. Il y a des années que les sociologues ont découvert que l'opinion moyenne d'un grand nombre d'observateurs (tous aussi experts, ou tous aussi ignorants) est d'un indicateur beaucoup plus fiable que l'opinion de l'un des observateurs, choisi au hasard. Ils ont appelé ce phénomène "l'effet de (l'oracle de)Delphes''. Il apparaît que Linus a fait la preuve que cela s'applique même au débogage d'un système d'exploitation -- que l'effet de Delphes peut apprivoiser la complexité du développement même au niveau de complexité atteint par le noyau d'un système d'exploitation.

Je dois à Jeff Dutky la remarque qu'on peut reformuler la Loi de Linus sous la forme "On peut paralléliser le débogage''. Jeff fait remarquer que, même si le débogage requiert que les débogueurs communiquent avec un développeur chargé de la coordination, une réelle coordination entre débogueurs n'étant pas indispensable. Ainsi, le débogage n'est pas la proie de cette augmentation quadratique de la complexité et des coûts d'organisation qui rend problématique l'ajout de nouveaux développeurs.

En pratique, le monde Linux n'est quasiment pas affecté par la perte théorique d'efficacité qui découle du fait que plusieurs débogueurs travaillent sur la même chose au même moment. L'une des conséquences de la politique du "distribuez tôt, mettez à jour souvent'' est de minimiser les pertes de ce type en propageant au plus vite les corrections qui sont revenues au coordinateur. Brooks a même fait une observation annexe proche de celle de Jeff: "Le coût total de maintenance d'un programme largement utilisé est typiquement de 40

pour cent ou plus de son coût de développement. De façon surprenante, ce coût dépend énormément du nombre d'utilisateurs. Un plus grand nombre d'utilisateurs trouve un plus grand nombre de bogues.'' (l'emphase est de mon fait).
Plus d'utilisateurs trouvent plus de bogues parce que l'ajout de nouveaux utilisateurs introduit de nouvelles manières de pousser le programme dans ses derniers retranchements. Cet effet est amplifié quand les utilisateurs se trouvent être des co-développeurs. Chacun d'entre eux a une approche personnelle de la traque des bogues, en utilisant une perception du problème, des outils d'analyse, un angle d'attaque qui lui sont propres. "L'effet de Delphes'' semble précisément fonctionner grâce à cette variabilité. Dans le contexte spécifique du débogage, la diversité tend aussi à réduire la duplication des efforts.
Ainsi, introduire de nouveaux bêta-testeurs ne va sans doute pas réduire la complexité du bogue le plus "profond'' du point de vue du développeur, mais cela augmente la probabilité que la trousse à outils d'un bêta-testeur sera adaptée au problème de telle sorte que le bogue saute aux yeux de cette personne.
Mais Linus assure ses arrières. Au cas où il y aurait des bogues sérieux, les versions du noyau Linux sont numérotées de telle sorte que des utilisateurs potentiels peuvent faire le choix d'utiliser la dernière version désignée comme étant "stable'', ou de surfer à la pointe de la technique courant le risque que quelques bogues accompagnent les nouvelles fonctionnalités. Cette tactique n'est pas encore formellement imitée par la plupart des bidouilleurs Linux, mais c'est sans doute un tort; le fait d'avoir une alternative rend les deux choix séduisants.

9. Prérequis nécessaires au style bazar

Les premiers relecteurs et les premiers publics auprès desquels cet article a été testé ont posé de manière régulière la question des prérequis nécessaires à la réussite d'un développement dans le style bazar, en incluant dans cette question aussi bien les qualifications du chef de projet que l'état du code au moment où il est rendu public et tente de rallier autour de lui une communauté de co-développeurs.
Il est assez évident qu'il n'est pas possible de commencer à coder dans le style bazar dès le début. On peut tester, déboguer et améliorer un programme dans le style bazar, mais il sera très difficile de faire démarrer un projet dans le mode bazar. Linus ne l'a pas tenté, moi non plus. Il faut que votre communauté naissante de développeurs ait quelque chose qui tourne, qu'on puisse tester, avec quoi elle puisse jouer.
Quand vous initiez un travail de développement en communauté, il vous faut être capable de présenter une promesse plausible. Votre programme ne doit pas nécessairement fonctionner très bien. Il peut être grossier, bogué, incomplet, et mal documenté. Mais il ne doit pas manquer de convaincre des co-développeurs potentiels qu'il peut évoluer en quelque chose de vraiment bien dans un futur pas trop lointain.
Linux et fetchmail furent tous deux rendus publics avec des conceptions simples, fortes et séduisantes. Nombreux sont ceux qui, après avoir réfléchi au modèle du bazar tel que je l'ai présenté ici, ont très correctement considéré que cela était critique, et en ont rapidement conclu qu'il était indispensable que le chef de projet fasse preuve au plus haut point d'astuce et d'intuition dans la conception.
Mais Linus se fonda sur la conception d'Unix. Ma conception provenait initialement du vieux programme popclient (bien que beaucoup de choses aient changé à terme, proportionnellement bien plus que dans Linux). Alors faut-il que le chef/coordinateur d'un effort commun mené dans le style bazar ait un

talent exceptionnel pour la conception, ou peut-il se contenter d'exalter le talent des autres ?
Je pense qu'il n'est pas critique que le coordinateur soit capable de produire des conceptions exceptionnellement brillantes. En revanche, il est absolument critique que le coordinateur soit capable de reconnaître les bonnes idées de conception des autres.
Les premières personnes auxquelles j'ai présenté cet article m'ont complimenté en me suggérant que je suis prompt à sous-évaluer dans la conception des projets menés dans le style bazar, leur originalité -- principalement parce que j'en suis pas mal pourvu moi-même, ce qui me conduit à considérer cela comme acquis. Il peut y avoir un fond de vérité là dedans; la conception (à l'opposé de la programmation ou du débogage) est certainement mon point fort.
Mais le problème avec l'intelligence et l'originalité dans la conception de logiciels, c'est que ça devient une habitude -- presque par réflexe, vous commencez à faire dans l'esthétique et le compliqué alors qu'il faudrait rester simple et robuste. Certains de mes projets n'ont jamais abouti à cause de cette erreur, mais ce ne fut pas le cas pour fetchmail.
Aussi suis-je convaincu que le projet fetchmail a réussi en partie parce que j'ai refoulé ma tendance à donner dans la subtilité; cela contredit (au moins) l'idée que l'originalité dans la conception est essentielle dans des projets réussis menés dans le style bazar. Et pensez à Linux. Imaginez que Linus ait tenté de mettre en pratique des innovations fondamentales dans la conception de systèmes d'exploitation au cours du développement; est-il probable que le noyau qui en résulterait soit aussi stable et populaire que celui que nous connaissons ?
Il faut, bien sûr, une certaine habileté dans la conception et le codage, mais je pense que quiconque envisage sérieusement de lancer un projet dans le style en possède déjà le minimum nécessaire. Les lois du marché de la réputation interne au monde du logiciel dont le code source est ouvert exercent des pressions subtiles décourageant ceux qui pensent mettre à contribution d'autres développeurs alors qu'ils n'ont pas eux-mêmes la compétence d'assurer le suivi. Jusqu'à présent tout cela semble avoir très bien marché.
Il existe une autre qualité qui n'est pas normalement associée au développement logiciel, mais je pense qu'elle est aussi importante qu'une bonne intelligence de la conception pour les projets de style bazar -- et elle est peut-être bien plus importante. Le chef, ou coordinateur, d'un projet dans le style bazar doit être bon en relations humaines et avoir un bon contact.
Cela devrait être évident. Pour construire une communauté de développement, il vous faut séduire les gens, les intéresser à ce que vous faites, et les encourager pour les petits bout du travail qu'ils réalisent. De bonnes compétences techniques sont essentielles, mais elles sont loin de suffire. La personnalité que vous projetez compte aussi.
Cela n'est pas une coïncidence que Linus soit un chic type qu'on apprécie volontiers et qu'on a envie d'aider. Cela n'est pas fortuit non plus que je sois un extraverti énergique qui aime animer les foules et qui se comporte et réagit comme un comique de théâtre. Pour que le modèle du bazar fonctionne, une petite touche de charme et de charisme aide énormément.

10. Le contexte social du logiciel dont le code source est ouvert

Cela est bien connu: les meilleures bidouilles commencent en tant que solutions personnelles aux problèmes de tous les jours rencontrés par leur auteur, et elles se répandent parce que ce problème est commun à de nombreuses personnes.
Dans "Le mythe du mois-homme'', Fred Brooks observa qu'on ne peut pas diviser et répartir le temps du programmeur; si un projet a du retard, lui

ajouter des développeurs ne fera qu'accroître son retard. Il explique que les coûts de communication et de complexité d'un projet augmentent de manière quadratique avec le nombre de développeurs, alors que le travail réalisé n'augmente que linéairement. Depuis, cela est connu sous le nom de "loi de Brooks'', et on la considère en général comme un truisme. Mais si seule la loi de Brooks comptait, Linux serait impossible.
Le classique de Gerald Weinberg "La psychologie de la programmation sur ordinateur'' apporta ce qu'on pourrait considérer après coup comme une correction vitale à Brooks. Dans sa discussion sur la "programmation non égoïste'', Weinberg observa que dans les boîtes où les programmeurs ne marquent pas le territoire de leur code, et encouragent les autres à y chercher les bogues et les améliorations potentielles, les améliorations sont drastiquement plus rapides qu'ailleurs.
Les termes choisis par Weinberg l'ont peut-être empêché de recevoir toute la considération qu'il méritait -- et l'idée que les bidouilleurs sur Internet puissent être décrits comme "non égoïstes'' fait sourire. Mais je pense que cet argument semble aujourd'hui plus vrai que jamais.
L'histoire d'Unix aurait dû nous préparer à ce que nous découvrons avec Linux (et à ce que j'ai expérimenté à une échelle plus modeste en copiant délibérément les méthodes de Linus): alors que l'acte de programmation est essentiellement solitaire, les plus grandes bidouilles proviennent de la mise à contribution de l'attention et de la puissance de réflexion de communautés entières. Le développeur qui n'utilise que son propre cerveau dans un projet fermé ne tiendra pas la route face au développeur qui sait comment créer un contexte ouvert, susceptible d'évoluer, dans lequel la traque des bogues et les améliorations sont effectuées par des centaines de gens.
Mais le monde traditionnel d'Unix n'a pas poussé cette approche dans ses derniers retranchements pour plusieurs raisons. L'un d'entre eux concernait les contraintes légales des diverses licences, secrets de fabrication, et autres intérêts commerciaux. Un autre (mieux compris plus tard) était que l'Internet n'était pas encore assez mûr.
Avant que les coûts d'accès à Internet ne chutent, il existait quelques communautés géographiquement compactes dont la culture encourageait la programmation "non égoïste'' à la Weinberg, et un développeur pouvait facilement attirer tout un tas de co-développeurs et autres touche-à-tout doués. Les laboratoires Bell, le laboratoire d'intelligence artificielle de l'institut de technologie du Massachusetts (MIT), l'université de Californie à Berkeley, devinrent les foyers d'innovations légendaires qui sont restés puissants.
Linux fut le premier projet qui fit un effort conscient et abouti pour utiliser le monde entier comme réservoir de talent. Je ne pense pas que cela soit une coïncidence que la période de gestation de Linux ait coïncidé avec la naissance du World Wide Web, ni que Linux ait quitté le stade de la petite enfance en 1993-1994, au moment où l'intérêt général accordé à Internet et que l'industrie des fournisseurs d'accès explosèrent. Linus fut le premier à comprendre comment jouer selon les nouvelles règles qu'un Internet omniprésent rendait possibles.
==========

Face à une innovation majeure qui accumule les succès, les sceptiques rétorquent toujours que la portée est limitée.
Il y a 15 ans, les gens disaient « les bricoleurs de la FSF (Free Software Foundation) ont développé quelques belles démonstrations mais rien de sérieux ni d’utilisable » et le projet GNU démontra le contraire. Les détracteurs dirent alors « le toolkit GNU est intéressant mais on est encore loin d’un système complet et opérationnel » et Linux fit son apparition. Aujourd’hui, les sceptiques prétendent encore qu’il ne faut rien attendre

au-delà des couches systèmes… Pourquoi devrions-nous croire ces pessimistes alors que le passé leur a toujours donné tort ?
Le fait est que, même pour des solutions très exigeantes comme les suites bureautiques, Open Office est désormais considérée comme une alternative viable à MS Office ou GIMP comme un concurrent de Photoshop sans parler de Firefox dont la popularité ne cesse de grandir !

Couche après couche, la progression se poursuit, aucun domaine n'est épargné (même pas les ERP…) !
Ceci dit, si on trouve des projets d'open source dans tous les domaines, force est de constater que les projets qui ont du succès sont tous regroupés autour d'outils pour informaticiens (Linux, GCC) développement de site Web (MySQL, Apache, GIMP). Donc, cette généralisation est encore très partielle et va prendre du temps, encore beaucoup de temps.
OpenOffice est considéré comme une alternative viable à MS-Office (sauf qu'il n'a toujours pas de correcteur grammatical), mais a très peu de parts de marchés. Même les particuliers veulent avoir MS-Office et sont prêts à payer pour ça.

Une autre grande question sur les projets Open Source est le nombre de contributeurs. On sait que Linux a eu un grand nombre de contributeurs mais quid des projets moins populaires ?
En dehors des développeurs payés par Sun, combien y a-t-il eu de collaborateurs externes pour MySQL ou Open Office ?
Les statistiques montrent que les programmeurs vraiment actifs sont peu nombreux, et que la plupart des contributions sont très simples (ce qui ne veut pas dire qu'elles ne soient pas précieuses : l'alarme qui indique une erreur est toujours bienvenue). Pour Apache par exemple, plus de 80 % des modifications proviendraient de seulement 15 programmeurs.

Il semble que Linus ai réussi à horizontaliser le système d'exploitation , un peu comme IBM a du le faire avec son PC. Linus a gardé la maintenance du noyau, mais la grande modularité de Linux a permit de distribuer le développement. Par exemple, les pilotes de périphériques ont pu être développés par des développeurs dans leur coin (par exemple, un développeur qui avait une carte graphique non supportée par Linux). X-Window a pu être développé séparémment du reste. Et Gnome au-dessus de X-Window.

Arrivé à ce point, on pourrait dire : « les développeurs aiment les projets à code source ouvert et alors ? Ce n'est pas pour autant que les clients, les vrais utilisateurs vont les adopter massivement ! ».
Détrompez-vous, cette adoption a déjà commencé et ce pour (principalement) deux raisons : la qualité (et donc la fiabilité) des projets Open Source est bien meilleure que celle des projets propriétaires (et ce résultat est justement bien expliqué dans l'article de Raymond) et la pérennité est sans commune mesure.
Une raison objective milite en faveur d'une pérennité supérieure des OSS face aux logiciels commerciaux : comme les OSS ne sont pas (et ne peuvent pas) être sous le contrôle d'une entité privée ou publique, il n'y a pas de danger de voir les projets arrêtés pour des raisons marketing ou politiques. Au contraire, les projets OSS naissent souvent pour assurer la pérennité d'un logiciel, l'histoire d'Apache en est une illustration.

Le gros avantage du logiciel libre est la possibilité d'avoir plusieurs entreprises qui supportent un même logiciel. C'est déjà le cas de Linux, avec Red Hat, Ubuntu, etc. Mais là encore il y a théorie et pratique et on commence à observer un dictat de Red Hat sur les logiciels d'entreprise pour Linux. Car si un logiciel pour Linux est sensé fonctionner sur toutes les distributions, les éditeurs de logiciels ne certifient souvent leurs logiciels que sur Red Hat. Les clients sont donc hésitant à utiliser une autre distribution. Si Linux a une cohésion au niveau du noyau, il n'y en n'a pas autant au niveau des librairies runtime, ce qui peut entrainer des incompatibilités.

Le concept du logiciel libre a été également utilisé par des consortiums (comme l'OS pour mobile Symbian). Là encore on assiste à un cas de logiciel créé par ses propres utilisateurs, même si les utilisateurs ne sont plus des particuliers mais des entreprises.

===
Apache HTTP Server, souvent appelé Apache, est un logiciel de serveur HTTP produit par l'Apache Software Foundation. C'est le serveur HTTP le plus populaire du Web. C'est un logiciel libre avec un type spécifique de licence, nommée licence Apache.
Apache est apparu en avril 1995. Au début, il s'agissait d'une collection de correctifs et d'additions au serveur NCSA HTTPd 1.3, qui était dans le domaine public en droit de la propriété intellectuelle français et le serveur HTTP alors le plus répandu. Quand le NCSA a cessé de supporter son serveur HTTP, ses utilisateurs se sont regroupés afin de partager les correctifs qui, jusque-là, circulaient de façon désorganisée.
De cette origine, de nombreuses personnes affirment que le nom Apache vient de a patchy server, soit "un serveur rafistolé". Par la suite, Apache a été complètement réécrit, de sorte que, dans la version 2, il ne reste pas de trace de NCSA HTTPd.
Au début, Apache était la seule alternative sérieuse et libre au serveur HTTP de Netscape. Depuis avril 1996, selon l'étude permanente de Netcraft, Apache est devenu le serveur HTTP le plus répandu sur Internet. La version 2 d'Apache possède plusieurs avancées majeures par rapport à la version 1, entre autres le support de plusieurs plates-formes (Windows, Linux et UNIX, entre autres), le support de processus légers UNIX, une nouvelle API et le support IPv6.
La fondation Apache (Apache Software Foundation ou ASF) a été créée en 1999 à partir du groupe Apache (Apache Group) à l'origine du serveur en 1995. Le serveur Apache est avec Linux la brique de base de l'ensemble appelé LAMP.
===

En revanche, dans la sphère commerciale, ce type de coup d'arrêt est chose courante (par exemple quand Sun décida de faire passer ses clients de SunOS à Solaris ou quand le même Sun racheta trois serveurs d'applications – Net Dynamics, Forté et Netscape ex-Kiva – sans être capable de choisir ensuite, les tuant finalement tous les trois !), au gré des changements de stratégies des éditeurs et constructeurs. Les clients sont très regardants sur la question de la pérennité des produits sur lesquels ils font reposer leurs développements, et on les comprend !
Or, il est de plus en plus clair qu'il n'y a pas de pérennité quand les acteurs (éditeurs, constructeurs) s'en mêlent, les seuls projets vraiment pérennes, ce sont les projets Open Source.
Aujourd'hui, ce sont les projets Open Source qui fournissent la base technologique sur laquelle les fournisseurs de logiciels vont s'appuyer de plus en plus (alors qu'auparavant, ce socle technologique était proposé par les fournisseurs leaders du marché, d'abord IBM puis Microsoft). Certains hébergeurs Web montrent l'exemple en offrant aujourd'hui des serveurs qui reposent sur LAMP (Linux, Apache, MySQL, PHP) et ils seront de plus en plus nombreux à privilégier cette plate-forme à cause de sa popularité et des avantages qu'elle présente.

Désormais, les éditeurs vont être de plus en plus souvent confrontés à un choix cornélien : soit s'ouvrir aux OSS et compenser les revenus des licences par le service (la part du service représente déjà la moitié des revenus d'IBM – à travers IBM Global Services –, et c'est l'activité qui croît le plus vite et qui est la plus rentable…), soit prendre le risque d'être progressivement marginalisés si l'évolution en faveur des logiciels à code source ouvert se confirme et s'amplifie… Et, tous les mois, on apprend que tel ou tel acteur décide de mettre le code de tel ou tel projet en Open Source, même dans les secteurs les plus en vue comme les systèmes d'exploitation pour mobiles :

après Androïd de Google c'est ua tour de Symbian de Nokia de rejoindre les rangs des projets à code source ouvert...

Microsoft et le raté de Vista

Au dernier exercice, clos fin juin (2008), le chiffre d'affaires de Microsoft a enregistré la première baisse de son histoire (-3%, à 58 milliards de dollars). «Le marché du PC est en pleine crise», se défend-on chez Microsoft. Peut-être, mais, dans le même temps, les ventes des Mac d'Apple ont crû de 15%. La raison de cette désaffection est plutôt à recher cher du côté de la division Windows, la seule à régresser significativement (-13%). Son célèbre logiciel, qui équipe 90% des ordinateurs dans le monde et représente 25% des revenus du géant américain, a connu un semi-échec avec Vista, l'édition précédente lancée en 2007. Trop lourde, bourrée de bogues et surtout incompatible avec un grand nombre d'applications, cette dernière avait été très mal accueillie, conduisant beaucoup de particuliers et d'entreprises à conserver leur ancienne version, Windows XP, sortie en 2001.

Aujourd'hui, on est habitué à ce que Microsoft soit le principal acteur du marché du logiciel mais il n'en est ainsi que depuis 1998. Avant cette année là, c'était encore IBM le plus gros fournisseur mondial de logiciels (certes, les ventes de logiciels IBM sont difficillement comparables à celles de Microsoft, Oracle et consorts puisque Big Blue vend surtout des solutions quasiment "obligatoires" et sur seulement sur ces propres plates-formes).

Lors de l'année 2000, Microsoft réalisa un chiffre d'affaires de $23 millards (avec 39000 salariés) alors qu'Oracle se contenta de $10 milliards (avec 43000 salariés). Un produit Microsoft typique coûte $200 et se vend à 10 millions d'exemplaires alors que Oracle obtiendra le même revenu avec 10 000 exemplaires d'un de ses logiciels à $200 000. Microsoft vend des biens de consommation alors qu'Oracle vend des biens d'équipement...

Mais le raté de Vista est plus profond qu'une histoire de version baclée. Il traduit un phénomène que Microsoft essaie de combattre depuis des années: la commoditisation de Windows. Ce dernier ne fait plus rêver. Certes, Vista était trop gourmand en ressources, avait une nouvelle interface graphique qui en a décontenancé plus d'un, avait des bugs et surtout beaucoup d'incompatibilités logicielles comme matérielles. Mais rappelons nous que ca avait également été le cas avec Windows 95. La grande différence est qu'en 1995 les consommateurs n'étaient pas satisfait de ce qu'ils avaient (Windows 3.11) et attendaient un changement. Ils se sont donc rués sur Windows 95 (certains se sont même mis sur des listes d'attentes) et lui ont pardonnés toutes ses faiblesses. 12 ans plus tard, les consommateurs sont en grande partie satisfait de ce qu'ils ont (Windows XP). Ce dernier n'a peut-être pas le sex-appeal de Mac OS X, mais il fait le travail qu'on lui demande. En d'autres termes, les consommateurs n'étaient plus enthousiasmé à l'idée de changer de système d'exploitation. Ils ont donc rejeté les défauts de Vista et ont été très irrités du fait que Microsoft les force à passer à Vista. On ne peut pas avoir le beurre, l'argent du beurre et le sourire de la crémière...

Microsoft semble avoir corrigé le tir avec Windows 7, mais le problème fondamental subsiste : il est de plus en plus difficile de convaincre le consommateur de l'utilité d'une nouvelle version de Windows. Et ce phénomène touche également MS-Office. MS-Office 2007 mis à part, les versions d'Office n'ont changé que très peu de choses pour les utilisateurs (Office 95, 97, 2000, XP et 2003). Certes, chaque version a apporté son lot de nouvelles fonctionnalités, mais étant donné que l'utilisateur moyen n'en n'utilise pas le 1/10ème...

Microsoft ne peut plus miser sur tous les chevaux

Microsoft l'a compris depuis longtemps: contrôler la plate-forme logicielle permet de contrôler l'industrie. Et l'une des tactiques de Redmond a été longtemps de miser sur tous les chevaux. Dans les années 80 la plate-forme logicielle se résumait au système d'exploitation du PC. Bill Gates a donc décidé d'être sur tous les fronts. Microsoft était ainsi derrière MS-DOS, co-développait OS/2 avec IBM, développait Windows de son côté et avait racheté les droits d'un Unix sur PC (commercialisé sous le nom de Xenix). Bill Gates a même tenté de former un partenariat avec Apple pour licencier MacOS, mais

Steve Jobs a décliné l'offre. A une époque où l'avenir du PC était encore incertain, Microsoft a misé sur le plus de chevaux possibles pour être sûr d'avoir le ticket gagnant.

Malheureusement pour Redmond, cette méthode commence à montrer ses limites du fait d'un marché informatique devenu trop complexe. Tout d'abord, il existe beaucoup trop de marchés à conquérir. On assiste en effet à une multiplication des plates-formes (Internet, mobile, etc.). Miser sur tous les systèmes d'exploitation possibles sur PC demandait beaucoup d'énergie. Miser sur toutes les plates-formes possibles actuelles en demande encore plus. Si bien qu'à l'heure actuelle il existe plusieurs marchés que Microsoft pourrait qualifier de stratégiques:

- Le navigateur Web
- La plate-forme Internet (les standards et formats utilisés sur le Web)
- La recherche Internet
- Les services Internet grand public
- Le cloud computing
- Les ordinateurs de poche (assistant numérique, smartphone, baladeur MP3)
- L'électronique de salon (console de jeu, magnétoscope numérique)

Cela fait beaucoup de marchés à conquérir. Si Microsoft était comme General Electric qui possède de nombreuses divisions dans des secteurs aussi variés que l'aviation ou la finance, il pourrait couvrir autant de marchés. Mais Microsoft est Microsoft et tout marché stratégique a besoin de l'implication du PDG, en l'occurence Steve Balmer. Steve Balmer a beau être quelqu'un de brillant, il ne peut pas suivre tous ces marchés à la fois avec la même énergie. Par exemple, Steve dédie un certain nombre d'heures de son temps à son équipe marketing qui décide comment les utiliser au mieux (une interview à tel magasine, une présentation à telle conférence). Bien évidemment, plus il y a de marchés à couvrir, plus le temps de Balmer est divisé. Si Microsoft a bien comme priorité affichée d'être le numéro un de la recherche Internet, Bill Gates comme Steve Balmer ont tous deux boudé les conférences dédiées à la recherche Internet ainsi que les experts du secteur. On a parfois l'impression que Redmond ne s'intéresse à certains marchés plus par contrainte que par désir réel.

Mais il existe encore plus de marchés à surveiller. Le nombre de marchés qui peuvent potentiellement être stratégiques a explosé. Dans les années 90, pris de surprise par le succès du Web et de Netscape, Microsoft s'est lancé tête baissée pour écraser ce dernier et imposer son navigateur Web... Et a du coup ignoré la monté de Google et l'importance de la recherche sur Internet. Maintenant que Redmond veut "tuer" Google (dixit son PDG), quel autre marché naissant va-t-il manquer ?
Il est en effet très difficile – voire impossible - de prédire l'avenir des marchés. Qui aurait pu prévoir qu'un jour Amazon.com se baserait sur sa puissante infrastructure informatique/Internet pour devenir un pionnier du cloud computing?

Parfois des marchés annexes apparemment anodins deviennent stratégiques du jour au lendemain. Microsoft s'est très tôt implanté sur le marché de l'assistant numérique, empêchant Palm de contrôler le secteur. Pas question de laisser à quelqu'un d'autre le contrôle de la plate-forme mobile qui pourrait un jour concurrencer l'ordinateur portable. Mais Bill Gates ne s'est pas rendu compte qu'Apple et son iPod - un vulgaire baladeur MP3 à ses yeux - pouvait s'attaquer à la dite plate-forme mobile. Ce que Steve Jobs ne s'est pas privé de faire en lançant l'iPhone puis l'iPod Touch qui sont des ordinateurs de poche bien plus puissants qu'un simple assistant numérique. Après avoir raillé l'iPod pendant des années et prédit un piètre avenir à ce type de produit, Microsoft a finalement décidé de concurrencer l'iPod en sortant le Zune (qui n'a pas été un succès, loin s'en faut !).
De la même manière, quel produit contrôlera l'électronique de salon ?
Est-ce que ce sera les consoles de jeux comme la Xbox ?
Une console plus familiale comme la Nintendo Wii ?
Le magnétoscope numérique ?
Ou quelque chose d'autre ?

Finalement, le dernier handicap de Redmond est son existant. Lorsqu'on a un chiffre d'affaire annuel de quelques $60 milliards, quelle doit être la taille d'une division pour exister :
$100 millions ?
$1 milliards ?
C'est à la fois pour cette raison (et en partie du fait de la culture même de Microsoft) que Redmond a sous-estimé certains marchés : l'absence présumée de revenu. Dans son livre "The Road Ahead" (1995) Bill Gates a complètement loupé l'avènement du Web. Il considérait en effet Internet comme un réseau où tout était gratuit. "Il n'y a pas d'argent à se faire. En quoi est-ce un business intéressant ?" a-t-il répondu à l'un de ses employés qui essayait de le convaincre de l'importance du Web. Et ce n'est que lorsque Google a commencé à dégager des bénéfices colossaux que Microsoft s'est vraiment intéressé à la recherche Internet.

Autre handicap du géant de Remond : une obsession de vouloir tout lier à Windows, même lorsque cela n'apporte pas grande valeur ajoutée. Là où le Kindle d'Amazon.com ou l'iPod Touch d'Apple peuvent se connecter directement à Internet sans passer par un ordinateur, Microsoft a une préférence à lier ses ordinateurs de poche à Windows Media Player. De même, certains modules de Windows Live demandent d'installer Virtual Earth 3D. Le résultat est certes joli, mais loin d'être aussi pratique qu'un site qui fonctionne quel que soit le navigateur et sans avoir à installer un plug-in.
Dans les années 80 certains affirmaient que Bill Gates connaissait ses concurrents mieux qu'ils se connaissaient eux-mêmes. Bill a en effet su très bien diriger Microsoft pour positionner ce dernier au bon endroit. Mais le monde a changé, et ni Bill Gates ni Steve Balmer ne peuvent plus garder les yeux sur tous les concurrents potentiels – personne ne le peut.

Le "SaaS" & "Cloud Computing", évolution probable pour les éditeurs de logiciels !
L'un des termes employés régulièrement ces temps-ci a été le Saas (Software as a Service) et le Cloud Computing. Avant d'aller plus loin, il est important de définir ses deux termes.
Le terms SaaS signifie que l'on n'achète pas un logiciel (ou plus exactement on achetait la licence d'un logiciel, licence qui est juste la concession d'un *droit d'utilisation*-révocable qui plus est...-, pas plus !) mais qu'on le loue (c'est aussi ce qu'on appelait les solutions ASP il y a quelques années pour *Applications Service Provider*).
Cette solution est de plus en plus en vogue. Et les éditeurs aiment l'idée, car cela leur permet d'assurer une rentrée d'argent régulière. Avec des systèmes hébergés en interne, le client avait toujours la possibilité d'annuler la maintenance. Avec le SaaS, plus possible.
Les avantages sont également importants pour les clients et cela explique le succès de la formule... Avant, avec les ventes de licences, c'est le client qui assumait le "risque technique" : c'était à lui d'installer le logiciel, de le paramétrer et de l'intégrer dans son existant (et on a vu avec SAP que cela pouvait représenter bien plus que le coût d'acquisition initial). Avec les solutions ASP, le risque technique est, sinon annulé, considérablement réduit : vous vous connectez à la solution Web de l'éditeur, vous faites votre paramétrage en ligne et ça roule...
Bon, il reste encore les inévitables difficultés d'intégration avec votre existant mais le fait de ne plus avoir à assurer l'installation (sans parler des mises à jour !) et l'exploitation représente déjà un énorme soulagement !
L'intérêt des solutions ASP n'est pas seulement technique, il est aussi commercial : avec la vente de licence, l'effort de l'éditeur se situait surtout en amont et, une fois votre chèque encaissé, il avait tendance à vous oublier jusqu'à la prochaine version... Là, le produit doit évoluer en permanence, les bugs doivent être vite corrigés et l'exploitation impeccable car vous pouvez "changer de crémerie" à tout moment. Certes, le verrouillage du client existe bien aussi dans le cas d'une solution ASP car vous n'avez pas envie de recommencer trop souvent l'investissement en temps que vous avez

consenti pour vous habituer au logiciel de l'éditeur mais il est bien plus réduit que dans le cas de la licence.

Le terme Cloud Computing est une résurgence de la notion d'*utility computing* dont on a parlé plus tôt, où c'est la puissance de calcul qui est louée, sur le modèle d'une compagnie d'électricité. La particularité est que le client paye en fonction de la puissance de calcul utilisée. Autrement dit, une entreprise peut augmenter *ainsi que diminuer* sa consommation en temps réel, sans avoir à renégocier de contrat. L'un des géants du cloud computing n'est autre qu'Amazon.com - oui, oui, le libraire en ligne. La firme de Jeff Bezos fournit en effet depuis 2006 une offre de *Cloud Computing* assez complète. Son offre EC2 (pour *Elastic Computer Cloud*) permet de louer autant de serveurs virtuels que l'on désire. Son service S3 (*Simple Storage Service*) permet de stocker autant de données que l'on désire. EC2 a ainsi été utilisé par certains journaux américains comme le Washington Post qui avait besoin d'une grande puissance de traitement pendant un temps relativement court (devoir convertir au plus vite 17000 pages en un format indexable). Le Washington Post a accompli son forfait en utilisant 200 serveurs virtuels en parallèle, pour un coût total de $144,62.

En quoi ces deux concepts sont-ils intéressant ?
Parce que les entreprises sur ce marché sont les rares entreprises qui s'adressent (entre autres) aux petits clients.

Gérer son système d'information a toujours été une solution ardue. Si bien qu'il existe de nombreuses technologies et produits qui se battent pour ce marché. Mais, comme d'habitude, la plupart des acteurs (en particulier les gros) visent les grosses entreprises. Le gros de la bataille se passe autour des solutions "*enterprise*", c'est-à-dire des solutions haut de gamme (et bien évidemment coûteuses). Les gros acteurs comme IBM ou Tibco semblent très peu intéressés à viser le marché des PME.

Or certaines offres de SaaS et de Cloud Computing visent précisemment le marché des petites entreprises. Le modèle SaaS est très adapté aux petites entreprises qui n'ont pas les ressources humnaines pour héberger et de maintenir leur système d'information. Et très adapté aux startups qui veulent lancer un nouveau service mais qui n'ont pas la main d'oeuvre nécessaire pour supporter l'installation de leur logiciel aux quatre coins du monde. Salesforce.com, le géant dans le domaine, propose une formule d'entrée de gamme pour son CRM à $25 par mois et par utilisateur.

Mais pour être crédible, une compagnie de SaaS doit fournir une infrastructure fiable : les problèmes de disponibilités sont bien moins pardonnés lorsque l'hébergement est sous-traité que lorsque les compagnies hébergent elles-mêmes leur système d'informations. C'est pour cela que de nombreuses compagnies de SaaS s'appuyent sur une solution de Cloud Computing fournie par une compagnie tierce. La compagnie de SaaS se focalise sur les services fournis au clients, et la compagnie de Cloud Computing se focalise sur la maintenance d'une infrastructure matérielle qui puisse monter en charge si besoin est, et qui ait le plus grande fiabilité possible. Avec ce modèle, une startup SaaS peut se payer une infrastructure de la même qualité qu'une grosse entreprise sans en payer le coût. Elle peut facilement faire face à des pics de demande *comme* à une baisse des besoins - elle ne paye que ce qu'elle utilise.

Cela vaut la peine de revenir sur le parcours de Saleforces.com fondée en 1999 par Marc Benioff, un ancien cadre d'Oracle (Larry Ellison a d'ailleurs fait partie des investisseurs initiaux dans ce qui était alors une start-up...). Très vite, Saleforces se distingue par son marketing agressif qui proclame "la fin du logiciel" et c'est la croissance de son service en ligne qui força ses concurents comme Siebel à offrir eux aussi une solution ASP en plus de leurs traditionnelles ventes de licences. On peut même affirmer que c'est le succès surprise de salesforce.com qui offrit un premier niveau de crédibilité aux solutions ASP. En effet, revenu à la mode depuis 1998, ce type de solution cherchait avidement sa première "success story" à mettre en avant pour permettre au secteur de prendre son

essor...

Ce qui est en train d'arriver avec la vague du SaaS et du Cloud Computing est sans doute la mutation inévitable qui va toucher les éditeurs de logiciels. Tout comme les constructeurs d'ordinateurs ont été obligés de passer de l'intégration verticale à l'intégration horizontale, les éditeurs sont touchés à leur tour : la vente de licence représente le passé et les solutions en ligne sont l'avenir de la profession.